Group key management for wireless networks

Teo, Joseph Chee Ming

2008

Teo, J. C. M. (2008). Group key management for wireless networks. Doctoral thesis,
Nanyang Technological University, Singapore.

<https://hdl.handle.net/10356/14968>

<https://doi.org/10.32657/10356/14968>

Group Key Management for Wireless Networks

Joseph Teo Chee Ming

School of Electrical & Electronic Engineering

A thesis submitted to the Nanyang Technological University
in fulfillment of the requirement for the degree of
Doctor of Philosophy

2008

© 2008 by Joseph Teo Chee Ming. All rights reserved.

To my parents

Acknowledgements

First and foremost, I would like to show my deepest gratitude and thank my supervisor Prof. Tan Chik How who has been an advisor, philosopher and guide to me. Throughout my four years of research, Prof. Tan has always been inspiring me, encouraging me and opened my eyes to the world of possibilities in research. He has spent countless number of hours teaching me on the basics of cryptography, how to work on interesting problems, write good papers and discover new problems and solutions. This Ph.D. thesis would not have been successful without his inspiration, encouragement, help and support. He has been an inspiration to me at Nanyang Technological University ever since I first met him during my undergraduate studies. I am also thankful to my other supervisors Prof. Ng Jim Mee and my A-star supervisor Prof Robert Deng for their help and support regarding my studies.

I would also like to thank Prof. Low Chor Ping, Head, Division of Information Communication Institute of Singapore (ICIS), the other professors in ICIS as well as other professors of the School of Electrical and Electronic Engineering (EEE) for their advices and suggestions during my Ph.D. studies. I would also like to thank Prof Colin Boyd of Queensland University of Technology for taking precious time during his busy schedule to have some discussions with me when he was in Singapore. I also wish to thank Dr. Yongdae Kim of University of Minnesota at Twin Cities, Dr. Cristina Nita-Rotaru of Purdue University and Alireza Hodjat of UCLA for providing the answers to some of the questions I posted to them regarding their work. Special thanks to Dr Mike Scott of Dublin City University for clarifying my questions on the Miracl software library. I would also like to express my thanks to my friend, Dr Raymond Choo for his advice and help during my Ph.D. studies. I would also like to express my gratitude to the anonymous reviewers for their valuable comments on my submitted papers.

I would like to thank A-Star, especially the Science and Engineering Research Council for

providing me with the A-Star Graduate Scholarship (AGS) that has financially supported me for my Ph.D. studies. Without this financial support, I might not have been able to embark on this Ph.D. studies and would also not have been able to go for several overseas conferences and meet with fellow researchers to gain new insightful thoughts and ideas.

I would also like to thank all the lab technicians and my colleagues at the Infocomm Research Lab for their friendship and support. In particular, I would like to thank Yang Xun, Xie Feng, Zhang Yan, Zhang Liang, Minh Ngoc, Xue Daojun, Fang Can and others for their time and help during our discussions.

My most grateful thanks to my parents, Teo Pwa Teck and Ong Chwee Lian for their invaluable love and support for me. My father has taught me to be creative and quick thinking and my mother has always been my greatest supporter. I would also like to thank my aunt and god-mother Mary and cousin Catherine for their love and encouragement to study hard during my Ph.D. studies. I am also grateful to my cousin Gary who has showed interest in my work despite his busy schedule and even given me many interesting and helpful suggestions for my research. I feel grateful for having them all in my life.

Table of Contents

Acknowledgements	iv
Summary	xi
List of Figures	xiv
List of Tables	xv
List of Abbreviations	xvi
List of Symbols	xix
Chapter 1 Introduction	1
1.1 Wireless Networks	2
1.2 Group Key Management	4
1.3 Challenges for Group Key Agreement in Wireless Networks	5
1.3.1 Authentication	6
1.3.2 Dynamic Topology in Wireless Networks	7
1.3.3 Scalability of Wireless Networks	7
1.3.4 Denial-of-Service Attacks	7
1.4 Motivation	8
1.4.1 Authentication	9
1.4.2 Dynamic Nature of Wireless Networks	9
1.4.3 Scalability	9
1.4.4 Denial-of-Service Attacks	9
1.5 Research Objectives	10
1.6 Major Contribution of the Thesis	10
1.7 Structure of the Thesis	12
Chapter 2 Preliminaries	15
2.1 Mathematical Background	15
2.1.1 Abstract Algebra and the Main Groups	15
2.1.2 Elliptic Curve Cryptography (ECC)	16
2.1.3 Bilinear Maps from Elliptic Curve Pairings	19
2.2 Computational Problems	19
2.2.1 Integer Factoring Problem	19
2.2.2 RSA Problem	20
2.2.3 Discrete Logarithm Problem	20

2.2.4	Computational Diffie-Hellman CDH Problem	20
2.2.5	Decisional Diffie-Hellman DDH Problem	21
2.3	Cryptographic Concept	21
2.3.1	Message Authentication Code	22
2.3.2	Cryptographic Hash Function	23
2.3.3	Encryption Scheme	23
2.3.4	Digital Signature Scheme	25
2.4	Wireless Networks	31
2.4.1	Infrastructured Wireless Networks	31
2.4.2	Infrastructureless Wireless Networks	32
2.4.3	Hybrid Networks	35
2.4.4	Inherent Characteristics of Wireless Networks	36
2.4.5	Heterogeneous and homogeneous Wireless Networks	38
2.5	Key Establishment Concept	39
2.5.1	Type of Adversary	40
2.5.2	Diffie-Hellman Key Agreement	40
2.5.3	Authentication and Key Exchange Protocols	41
2.6	Group Key Agreement Protocols	43
2.6.1	Security Goals for Group Key Management	44
2.6.2	Static and Dynamic Group	45
2.7	Examples of Group Key Management Protocol	46
2.7.1	ING Protocol	46
2.7.2	Hypercube and Octopus Protocols	46
2.7.3	Group Diffie-Hellman Protocol	47
2.7.4	Tree-Based Group Diffie-Hellman Protocol	49
2.7.5	Skinny Tree Protocol	51
2.7.6	Burmester-Desmedt Group Key Agreement Scheme	52
2.7.7	Nam et al DDH-based Group Key Agreement Protocol	53
2.7.8	Nam et al Dynamic Group Key Exchange Protocol	53
2.7.9	Nam's Hierarchical Scheme	56
2.7.10	Saeednia and Safavi-Naini scheme	57
2.7.11	Kim-Lee-Lee Group Key Agreement Protocols	58
2.7.12	Dutta and Barua's Dynamic Group Key Agreement Protocol	61
2.7.13	Other GKA protocols in Current Literature	64
2.7.14	Password-based GKA Protocols in Current Literature	67
2.7.15	Abdalla et al's Password-based GKA Protocol	68
2.7.16	Byun et al's EKE-D Protocol	69
2.8	Summary on the General Idea of GKA Protocols	70
2.8.1	Non-controller based GKA Protocols	71
2.8.2	Controller based GKA Protocols	72
2.9	Summary of GKA Protocol Topologies	74
2.9.1	Bus topology	74
2.9.2	Tree Topology	74
2.9.3	Ring Topology	75
2.9.4	Star Topology	75
2.10	Attacks on Group Key Management Protocols	75
2.11	Denial-of-Service Attacks and Countermeasures	77

2.12	Summary of Hierarchical Group Model Topologies	79
2.12.1	Generalized Circular Hierarchical (C-H) Group Model	79
2.12.2	Contributory Ring-Centralized Group Model	82
2.12.3	Centralized-Contributory Ring Group Model	84
2.12.4	Centralized-Centralized Group Model	85
Chapter 3	Energy Analysis	87
3.1	Introduction	87
3.2	Current Work in Literature	88
3.3	Computational Energy Cost Analysis	90
3.3.1	133MHz SA-1110 “StrongARM” Microprocessor	90
3.3.2	206 MHz SA-1110 “StrongARM” Microprocessor	94
3.4	Communication Energy Cost Analysis	95
3.4.1	100kbps transceiver module and IEEE 802.11 Spectrum24 LA-4121 WLAN card	96
3.4.2	Lucent IEEE 802.11 11Mbps WaveLAN card	98
3.4.3	Certificate Size	99
3.5	Summary	101
Chapter 4	Authenticated Group Key Agreement for Wireless Networks	102
4.1	Introduction	102
4.2	Characteristic of Burmester-Desmedt protocol (BD)	103
4.3	Authentication Techniques for Burmester-Desmedt (BD)	104
4.3.1	Authenticated BD using digital signatures	104
4.3.2	Authenticated BD using an extra round	105
4.4	Proposed Authenticated Group Key Agreement Protocol	106
4.4.1	Security Analysis	108
4.5	Comparison of Computational Complexity	111
4.6	Summary	112
Chapter 5	Authenticated Dynamic Group Key Agreement for Autoconfigurable MANETs	113
5.1	Introduction	113
5.2	Cryptanalysis of Steiner et al. Join Protocol for BD	115
5.3	Cryptanalysis of Dutta and Barua’s Leave Protocol	117
5.3.1	Attack on the Dutta and Barua’s Leave Protocol	117
5.3.2	Flaw in Dutta and Barua’s security proof	118
5.4	Authenticated Dynamic GKA for Autoconfigurable MANETs	119
5.4.1	IP Address Autoconfiguration Classes	119
5.4.2	Requirements	119
5.5	The Proposed Protocols	120
5.5.1	Initial Key Agreement (Group Formation) Protocol	121
5.5.2	Join Protocol	122
5.5.3	Merge Protocol	124
5.5.4	Leave Protocol	127
5.6	Security Analysis	129
5.6.1	Security Analysis of IKA Protocol	129
5.6.2	Security Analysis of Join Protocol	130

5.6.3	Security Analysis of Merge Protocol	132
5.6.4	Security Analysis of Leave Protocol	134
5.7	Complexity Analysis	136
5.8	Discussion	138
5.9	Summary	140
Chapter 6 Energy-Efficient ID-based Group Key Agreement Protocols for Wire-		
	less Networks	141
6.1	Introduction	141
6.2	Proposed ID-based Authenticated GKA Protocol	142
6.2.1	Initial Group Key Agreement (IKA) Protocol	142
6.2.2	Join Protocol	144
6.2.3	Merge Protocol	145
6.2.4	Leave Protocol	147
6.3	Security Analysis	148
6.3.1	Security Analysis of IKA Protocol	148
6.3.2	Security Analysis of Join Protocol	150
6.3.3	Security Analysis of Merge Protocol	151
6.3.4	Security Analysis of Leave Protocol	153
6.4	Complexity Analysis	154
6.5	Energy Analysis	156
6.6	Summary	159
Chapter 7 Energy-Efficient and Scalable Group Key Agreement for Large Ad		
	Hoc Networks	160
7.1	Introduction	160
7.2	Group Key Agreement Protocol for Circular Hierarchical (C-H) Group Model	161
7.2.1	Security Analysis	163
7.3	Complexity Analysis	164
7.4	Energy Analysis	166
7.5	Summary	170
Chapter 8 Low-Power Group Key Agreement for Heterogeneous Wireless Net-		
	works	171
8.1	Introduction	171
8.2	GKA Protocol for ContRi-Central Group Model	172
8.3	Security Analysis	175
8.4	Complexity Analysis	178
8.5	Energy Analysis	181
8.5.1	Energy Consumption Cost Results	182
8.5.2	Discussion	187
8.6	Summary	188
Chapter 9 Denial-of-Service Attack Resilience Dynamic Group Key Agreement		
	for Heterogeneous Networks	189
9.1	Introduction	189
9.1.1	Certification Authority	190
9.1.2	Heterogeneous Network nodes	190

9.2	Proposed Group Key Agreement Protocols for the Central-ContRi Group Model . .	191
9.2.1	Initial Group Key Agreement	192
9.2.2	Join Protocol	196
9.2.3	Leave Protocol	201
9.3	Detection of Malicious Group Members	205
9.4	Security Analysis	207
9.4.1	Security Analysis of Initial GKA Protocol	207
9.4.2	Security Analysis of Join Protocol	213
9.4.3	Security Analysis of Leave Protocol	218
9.5	Complexity Analysis	224
9.6	Energy Analysis	228
9.6.1	Energy Consumption Cost Comparison Results	228
9.7	Summary	232
Chapter 10	Denial-of-Service Password-based GKA for Wireless Networks . . .	233
10.1	Introduction	233
10.2	Security Analysis and Improvement of a DoS-Resistant Authentication and Key Exchange Protocol	236
10.2.1	Attacks on Wan et al's Authentication and Key Exchange Protocol	236
10.2.2	Proposed Authentication and Key Exchange Protocol	239
10.2.3	Security Analysis of proposed authentication and key exchange protocol . . .	241
10.2.4	Complexity Analysis of proposed authentication and key exchange protocol .	244
10.3	Denial-of-Service Resilience Password-based Group Key Agreement Protocols for Wireless Networks	244
10.3.1	Security Analysis	257
10.3.2	Complexity Analysis	273
10.3.3	Energy Analysis	275
10.4	Summary	278
Chapter 11	Conclusion and Recommendations	279
11.1	Conclusion	279
11.2	Open Problems and Future Directions	283
Author's Publications		285
Bibliography		287

Summary

With the advancement of wireless networks in recent years, group communication via wireless networks is becoming an increasingly popular research area due to the growth of group oriented applications such as video conferencing, pay-per-view media distribution, real-time information services and collaborative work in wireless networks. Besides civilian applications mentioned above, group-oriented applications and protocols are also used widely in military operations. However, group communication via wireless networks takes place over radio waves, the common mode of communication in wireless networks, causes major security concern as radio transmission can be easily eavesdropped. Solutions such as IPSec that provides end-to-end secure communications between two communicating parties and user nodes would require many different secret keys to communicate with every other node in the network. A better solution is to use a shared secret group key for secure group communications in a wireless network. To achieve this, the nodes in the wireless network will first run a group key management (GKM) protocol to establish a common group key. This key will then be used as a secret key for symmetric-key cryptography to encrypt messages that they wish to broadcast to the wireless network. In this way, entities outside the wireless network will not be able to decode these messages.

Besides being secure and efficient, the proposed GKM protocol should also be authenticated. Authentication is necessary in GKM protocols to prevent adversaries from masquerading as group members and obtain the group key. Besides authentication, the inherent characteristics of wireless networks such as high dynamic topology and computational power of network nodes (e.g. low power energy constrained devices) have to be taken into account when designing GKM protocols for wireless networks. Furthermore, different types of wireless networks will affect the design of the GKM protocol. For example, the GKM protocol designed for wireless mesh networks (WMNs) or wireless local area networks (WLANs) can have controllers to lower the burden of user nodes since

the mesh routers and access points in such networks can take the role of controllers.

There are already many GKM algorithms/protocols in current literature. However, some critical issues, such as energy consumption, communication costs, scalability, protection against Denial-of-Service (DoS) attacks, etc., have not been well understood and need to be deeply studied. In this Ph.D. thesis, we address four challenges in the design of GKM protocols for wireless networks.

1. How to design more efficient authentication techniques for GKM protocol in different types of wireless networks?
2. How to design efficient and secure dynamic GKM protocols to handle the highly dynamic topology of certain types of wireless networks?
3. How to handle the scalability issue when executing GKM for large-scale wireless networks?
4. How to tackle the problem of DoS attacks against GKM protocols for wireless networks?

This thesis is dedicated to the solutions of the above problems.

First, we identify an interesting characteristic of an efficient, secure and popular group key agreement (GKA) protocol (the Burmester-Desmedt (BD) protocol) that is very suitable for wireless networks. Next, we design an efficient authenticated GKA protocol based on this characteristic of the BD protocol. The complexity analysis indicates that the proposed protocol is more efficient than current methods to authenticate the BD protocol.

Next, we look into the problem of designing dynamic GKM protocols for wireless networks. As mentioned earlier, the BD is a popular and efficient GKA protocol and several research works have been proposed to extend the BD to handle dynamic events such as user join and user leave. However, we show that without careful analysis and study, security flaws can occur. We first show a security flaw in a method briefly mentioned by Steiner *et al.* to provide a Join protocol for the BD. Next we show another security flaw in the provable-secure dynamic GKA protocol of Dutta and Barua. We then proposed three authenticated GKA protocols namely Join, Leave and Merge protocols for mobile ad hoc networks (MANETs). Our proposed protocols provide autoconfiguration which is useful when implementing MANETs in application. The complexity analysis of our proposal with four current dynamic GKA protocols show that our proposed protocols are efficient and suitable for MANETs.

Next, we explore both the authentication issue and the highly dynamic topology of wireless networks by investigating ID-based authentication. We proposed an ID-based authenticated Initial GKM protocol and three ID-based authenticated dynamic GKM protocols, namely Join, Leave and Merge Protocols. Besides complexity analysis, we also provide an energy analysis of our proposed protocols that show that our proposed protocols are more efficient and the energy consumed by nodes running our proposed protocols is significantly lower.

To focus on the problem of scalability in wireless networks, especially large-scale wireless networks, we developed different hierarchical group models and designed GKM protocols that use these different hierarchical group models for different types of wireless networks. The complexity and energy analysis of our GKM protocols that use the various hierarchical group models show that they are all more scalable and energy-efficient than current GKM protocols including current hierarchical GKM protocols.

Finally, we look into the problem of DoS attacks on GKM protocols in wireless networks. Most GKM protocols in current literature do not consider DoS attacks even though DoS attacks are very serious threats to real-life implementation of wireless networks as it is fairly easy for the attacker to launch DoS attacks. We propose GKM protocols that not only provide protection against DoS attacks but also are energy efficient and scalable. One of the proposed DoS-resistant GKM protocol uses password-based authentication, which is a better alternative to the current technique of using digital signatures. We also show three different DoS attacks against an anonymous access control protocol [WDBA05] that claims to be DoS resistant.

List of Figures

2.1	Elliptic curve $y^2 = x^3 - 8x + 6$	17
2.2	Hypercube protocol for 4 users	47
2.3	Octopus protocol for 13 users	47
2.4	Key Tree in TGDH	49
2.5	A five member STR Key Tree	51
2.6	Nam's Hierarchical Scheme with n users	56
2.7	Bus Topology	74
2.8	Tree Topology	74
2.9	Ring Topology	74
2.10	Star Topology	74
2.11	An illustration of the (C-H) Group Model with $c = 10$	79
2.12	User nodes in the (C-H) Group Model with $c = 3$ and $h = 3$	81
2.13	A ContRi-Central Group Model.	82
2.14	Nodes in the ContRi-Central Group Model with $c = 3$ and $h = 3$	84
2.15	The Centralized-Contributory Ring (Central-ContRi) Group Model.	85
2.16	Centralized-Centralized Group Model.	86
3.1	Extrapolation	93
3.2	X.509 (Version 3) certificate	100
6.1	Energy Consumption Costs.	157
7.1	Computational Energy Consumption Costs.	167
7.2	Transmission (Communication) Energy Consumption Costs.	167
7.3	Reception (Communication) Energy Consumption Costs.	167
7.4	Total (Computational and Communication) Energy Consumption Costs.	167
7.5	Total Energy Consumption by each GKA protocol.	170
8.1	Computational Energy Consumption Costs.	184
8.2	Transmission Energy Consumption Costs.	184
8.3	Reception Energy Consumption Costs.	185
8.4	Total Energy Consumption Costs.	186
8.5	Total Energy Consumption Costs for each GKA.	187
9.1	Energy Consumption Cost of user nodes for Initial GKA protocol.	229
10.1	Energy Costs of user nodes for Initial Group Key Agreement (IKA) Protocol.	276

List of Tables

2.1	Timing Costs for RSA and DSA signature scheme on a 3GHz Pentium IV system . .	29
3.1	Costs of Cryptographic Operations on 133MHz StrongARM microprocessor and P3-450MHz microprocessor	92
3.2	Computational Energy Cost for 133MHz SA-1110 “StrongARM” Microprocessor . .	94
3.3	Computational Energy Costs for 206MHz SA-1110 “StrongARM” Microprocessor . .	95
3.4	Communication Energy Cost for 100kbps transceiver module and IEEE 802.11 Spectrum24 LA-4121 WLAN card	98
3.5	Communications Energy Costs for Lucent IEEE 802.11 11Mbps WaveLAN card . . .	99
4.1	Complexity Analysis	111
5.1	Computation and Communication Costs	136
6.1	Complexity Analysis for Authenticated BD Protocol	155
6.2	Complexity Analysis of Dynamic Protocols	156
6.3	Energy Cost for Dynamic Protocols	158
7.1	Computation and Communication Cost	165
8.1	Computational and Communication Cost	179
9.1	Computational and Communication Costs	226
9.2	User Energy Costs for Join Protocol(20 Users)	230
9.3	User Energy Costs for Leave Protocol (20 Users)	231
10.1	Message Size in bits	244
10.2	Computational and Communication Cost	274
10.3	User Energy Costs for Join Protocol(20 Users)	277
10.4	User Energy Costs for Leave Protocol (20 Users)	278

List of Abbreviations

AP	Access Point
AAC	Anonymous Access Control
AES	Advanced Encryption Standard
BD	Burmester-Desmedt Protocol
BS	Base Station
CA	Certificate Authority
CBC-MAC	Cipher-Block-Chaining Message Authentication Code
CPA	Chosen Plaintext Attack
CCA	Chosen Ciphertext Attack
C-H	Circular-Hierarchical
CG	Control Group
CDH	Computational Diffie-Hellman
DES	Data Encryption Standard
DDoS	Distributed Denial of Service
DoS	Denial of Service
DH	Diffie-Hellman
DDH	Decisional Diffie-Hellman
DSA	Digital Signature Algorithm
DL	Discrete Logarithm
EAP	Extensible Authentication Protocol
ECC	Elliptic Curve Cryptography
ECDSA	Elliptic Curve Digital Signature Algorithm

EKE-D	Byun et al's password-based Group Key Agreement Protocol
ESTAR	Variant of the VSTAR protocol
gcd	greatest common divisor
GDH	Group Diffie-Hellman
GKA	Group Key Agreement
GKM	Group Key Management
GQ	Guillou and Quisquater ID-based Signature Scheme
GSM	Global System for Mobile Communications
HMAC	Hashed Message Authentication Code
HWN	Heterogenous Wireless Network
ID	Identity
IBE	Identity-based Encryption
IKA	Initial Group Key Agreement
IND	Indistinguishability of Encryption
ING	Ingemarsson Group Key Agreement Protocol
IP	Internet Protocol
IPSec	IP Security
MANET	Mobile Ad hoc Network
MD5	Message Digest 5
MACS	Message Authentication Code Scheme
NHS	Nam's Hierarchical Scheme
NLKW	Nam et al's DDH-based Group Key Agreement
NLKKW	Nam et al's Dynamic Group Key Exchange
PAN	Personal Area Network
P2P	Point-to-point
PDA	Personal Digital Assistant
PKC	Public Key Cryptography
PKE	Public Key Encryption Scheme
PKI	Public Key Infrastructure

PKG	Private Key Generator
RADIUS	Remote Authentication Dial In User Service
RSA	Rivest-Shamir-Adleman
SHA	Secure Hash Algorithm
SKE	Symmetric Key Encryption Scheme
SOK	Sakai-Ohgishi-Kasahara's ID-based Signature Scheme
SSL	Secure Socket Layer
SSN	Saeednia and Safavi-Naini
STR	Skinny Tree
TAP	Transit Access Point
TGDH	Tree Group Diffie-Hellman
TTP	Trusted Third Party
UAV-MBN	Unmanned Aerial Vehicle-Mobile Backbone
VBD	Variant of the BD protocol
VK-GKA	Variant of the Kim's GKA protocol
VSTAR	Variant of the NLKW Protocol
VPN	Virtual Private Network
WEP	Wired Equivalent Privacy
WHS	Wireless Hot Spot
WINS	Wireless Integrated Network Sensor
Wi-Fi	Industry name for Wireless Local Area Network
WLAN	Wireless Local Area Network
WMAN	Wireless Metropolitan Area Network
WMN	Wireless Mesh Network
WSN	Wireless Sensor Network
XOR	Exclusive Or

List of Symbols

$E_X(m)$	Public key encryption of message m using X's public key
$\mathcal{E}_K(m)$	Symmetric key encryption of message m using key K
$\mathcal{M}_K(m)$	Message authentication code generation of message m using key sk
$\mathcal{S}_{sk}(m)$	Digital signing of message m using key sk
$H(\cdot), H_1(\cdot), H_2(\cdot)$	Cryptographic hash functions
$H_K(\cdot)$	Cryptographic keyed hash function using key K
$\parallel$	Concatenation
n	group size
$\mathcal{L}_v$	v th layer for $0 \leq v \leq h - 1$ in the hierarchical group model
h	height/total number of layers in hierarchical group model
c	number of members in each subgroup
n_c	number of subgroups
CG	Control Group
$\mathcal{G}_{v,j}$	j th subgroup at $\mathcal{L}_v$ for $0 \leq j \leq c^v - 1$
$\mathcal{G}_i$	Subgroup $\mathcal{G}_i$ for $1 \leq i \leq n_c$
n_i	number of members in subgroup $\mathcal{G}_i$
C	Main Controller
U_i	i th group member/user
$S_{v,j}$	Subgroup controller of subgroup $\mathcal{G}_{v+1,j}$ for $0 \leq v \leq h - 2$ and $0 \leq j \leq c^{v+1} - 1$
$S_{h-1,i}$	i th user at the lowest layer $\mathcal{L}_{h-1}$ for $0 \leq i \leq c^h - 1$
S_i	Subgroup Controller of subgroup $\mathcal{G}_i$
$U_{i,j}$	j th user in subgroup $\mathcal{G}_i$ for $1 \leq j \leq n_i$

$K_{v,j}$	Subgroup key of subgroup $\mathcal{G}_{v,j}$
K_i	Subgroup key of subgroup $\mathcal{G}_i$
K	Final group key
$\text{addr}[\mathcal{U}_i]$	IP address of node $\mathcal{U}_i$
$\text{cert}[\mathcal{S}_{v,j}]$	certificate of node $\mathcal{S}_{v,j}$
σ_i	signature generated by $\mathcal{U}_i$
SID_i	Session ID of subgroup $\mathcal{G}_i$
$\mathcal{S}$	Authentication Server
$\mathcal{C}$	Client
AP	Access Point
MAC_X	Media Access Control (MAC) address of X
IP_X	Internet Protocol (IP) address of X
pw	password shared between client & server

Chapter 1

Introduction

Over the years, wireless networks have become an important part of our daily life. The use of cellular phones as a means of communication is one example that wireless networks have become an integral part of our society. With the wide availability of current wireless technologies such as Bluetooth, IEEE 802.11, 2G and 3G and future wireless technologies such as WiMax, wireless mesh networks and mobile ad hoc networks (MANETs), more and more applications that exploit the advantages of different types of wireless networks such as wireless ad hoc networks and wireless mesh networks (WMNs) are being envisioned. Group communication in wireless network has become important due to the growing popularity of group oriented applications such as video conferencing, pay-per-view media distribution, real-time information services and collaborative work in wireless networks. Besides civilian applications, group-oriented applications and protocols are also used widely in military operations. For example in a military network, a battalion could be considered as a large group and soldiers in the battalion are required to communicate with one another. In this case, a wireless network (e.g. ad hoc network) could be set up for group communications within the battalion.

However, messages broadcast in wireless networks are transmitted in plain radio waves. Therefore, anyone can easily listen to the messages being broadcast [BGW01]. Furthermore, many of these group communication applications in wireless networks distribute sensitive information. Hence, there is a need to maintain the confidentiality and integrity of the data exchanged dur-

ing the group communication. One straightforward solution is the use of public key cryptography (PKC). That is, when a sender **A** wishes to send a message to receiver **B**, **A** will use **B**'s public key to encrypt the message before sending to **B**. Upon receiving the message, **B** will use its private key to decrypt and obtain the actual message. However, studies [CKM00,HV02] have shown that the energy requirements of PKC are orders of magnitude *higher* than symmetric-key cryptography. Hence, it is more efficient to use symmetric-key cryptography instead of public-key cryptography for secure group communication in wireless networks. This is especially desirable and useful for wireless networks since most user devices are battery-operated, and there is a need to prolong their battery lifespan. One widespread solution of using symmetric-key cryptography for secure communication is to first employ a two-party key exchange protocol such as IPSec [DH03,FKL⁺05] to establish a key between two communicating parties so as to create a Virtual Private Network (VPN) in a tunnel for secure communications between the two communicating parties. However, this solution requires each node to have different secret keys to communicate with every other node in the wireless network. Furthermore, VPN solutions can only handle unicast traffic. A better solution is the use of a shared secret group key for secure wireless group communications. To achieve this, the nodes in the wireless network will first run a group key management protocol to establish a common group key. This key will then be used as a secret key for symmetric-key cryptography to encrypt messages that they wish to broadcast to the wireless network. Entities outside the network would not be able to decode these messages.

1.1 Wireless Networks

Wireless networks can be classified into three categories, namely infrastructured, infra-structureless and hybrid.

Infra-structured network These networks consist of mobile devices that communicate with each other through one or more central devices, called base stations (BSs) or access points (APs). Examples of this type of networks include the cellular telephone network where each phone communicates with a BS and not directly with another telephone. Another example is the wireless LAN (i.e. IEEE 802.11b), where there are APs that handles the communications of the user nodes.

Infra-structureless network These networks are characterized by the lack of centralizing devices such as APs or BSs. The user devices in these networks are able to communicate directly with other devices that are within its communication range. Infra-structureless wireless networks are also known as ad hoc networks. Ad hoc networks can be formed “on the fly” and are thus useful in environments that do not have any network infrastructures. Examples of infra-structureless environments include places of difficult access (e.g. mountains, forests, deserts, etc) and emergency situations (e.g. flooding, earthquake, rescue operations etc). Depending on the scenario that the ad hoc network is created in, the devices can be assumed to be static, have little mobility or are highly mobile. For example, in a conference room, a group of executives could set up an ad hoc network for an electronic meeting. The devices in this type of ad hoc network setting can be assumed to be static or have little mobility. Multi-hop ad hoc networks are ad hoc networks in which all devices help to route packets to other devices in the network. Hence, two devices that are not within communicating range of one another can still communicate with each other if there is at least one chain of devices that is reachable by both. These chain of devices will help to route the packets through multiple hops to its destination. Wireless sensor networks (WSNs) [ASSC02] are another example of infra-structureless networks. In WSN, hundreds or thousands of nodes could be deployed to collect data on the environment (e.g. enemy movement in military operations etc).

Hybrid network Hybrid networks consist of elements from both infrastructure and infrastructureless networks. For example, the backbone domain of a hybrid network could be an infrastructure network which consists of servers, APs and BSs while the user domains are either infrastructure or infrastructureless networks. In general, the server/controller nodes that form the backbone are typically high power nodes while the user devices in the user domains are low power devices such as Personal Digital Assistants (PDAs) and laptop computers.

One typical example of a hybrid network is the Wireless Mesh Network (WMN) [AWW05] . WMNs are typically large-scale networks where the backbone comprises of mesh routers (servers or controllers) and the user domains may contain infrastructured networks such as Wi-Fi or cellular networks or infrastructureless networks such as ad hoc networks or sensor networks connected to the backbone domain via a mesh router. Another example of hybrid network is the military network known as Unmanned Aerial Vehicle-Mobile Backbone Node (UAV-MBN) network [CL06].

In UAV-MBN networks, the Mobile Backbone Nodes (MBNs) are high power devices such as army tanks or jeeps that form the infrastructured part of the network while the low-power user nodes are typically soldiers carrying handheld devices such as PDAs and laptop computers powered by batteries. These low-power user nodes (soldiers) are split up into small groups or subgroups (e.g. platoons or section in the army) and form the infra-structureless part of the UAV-MBN network.

Characteristics of Wireless Networks User nodes of wireless networks (e.g. PDAs) are usually bandwidth-constrained and energy-constrained. Therefore, computations must be kept at minimum and transmission range may be low due to lack of power. Nodes in an mobile wireless network may also tend to move in and out of range of other nodes frequently. This may cause links to break and the network topology to change at any time. Infrastructured or hybrid networks where there are user nodes with limited computing capabilities and powerful nodes without energy constraints are also known as **asymmetric** wireless networks while infra-structureless networks where all nodes have equal limited computing capabilities are known as **symmetric** wireless networks.

1.2 Group Key Management

Group Key Management refers to a set of techniques and procedures to support the establishment and maintenance of group keys between authorized parties in a group. The established group key can be used for symmetric key encryption to provide secure group communications. There are three different approaches to Group Key Management (GKM), namely centralized group key distribution, decentralized group key distribution and contributory group key agreement.

Centralized group key distribution This approach relies on a single entity (also known as the key server) which could be one of the group members or an external Trusted Third Party (TTP) to generate and distribute the group keys. The key server maintains long-term shared keys with each group member and uses these shared keys to distribute the group key securely to all group members. Although this type of key distribution minimizes the computational power of the users and provides bandwidth utilization, it has two major drawbacks. The first is that the user designated to be the key server would require greater computational power in order to distribute

the group key to all group members. On the other hand, if the key server is not a group member but a TTP, then this TTP would have to be trusted by the group. The second drawback is that the key server may generate weak keys that lessen the security of the group communication.

Decentralized group key distribution In this architecture, the management of the whole group (the role of the key server) is distributed to a number of group members, known as subgroup controllers. The entire group is split into smaller subgroups and different controllers are used to manage each subgroup. This reduces the chances of the key server creating weak keys and lessens the computational weight on the key server. Furthermore, more entities are allowed to fail before the whole group is affected.

Contributory group key agreement This approach is characterized by having all users in the group contribute to a portion of the common group key. This approach avoids the problem of centralized trust and single point of failure. It also reduces the risk of malicious behavior. All members can perform access control. In addition, contributory group key agreement protocols require each user to be aware of the group membership list to ensure that the protocols are robust.

Due to above-mentioned advantages of contributory group key agreement over centralized and decentralized group key distribution, we focus on contributory group key agreement in this thesis. In this thesis, we shall use the terms group key agreement and group key management synonymously.

1.3 Challenges for Group Key Agreement in Wireless Networks

Contributory group key agreement (GKA) for wireless networks not only have to ensure that the basic security requirements for traditional GKA are met, but they also have to consider the characteristics of the wireless network when designing an effective and efficient GKA protocol for wireless networks. For example, user nodes in wireless networks are typically low power energy constrained devices such as PDAs that have lower computational capabilities compared to an average desktop system in wired network. Furthermore, user devices in wireless networks are usually powered by batteries which are limited resource. Moore's Law [Moo98] states that the number of transistors in a microprocessor will double approximately every 18 months (i.e. the power of

computing will grow exponentially). However, the capacity of batteries only grows linearly and as a result, there is a difference (“power gap”) between the power required by computing and battery capacity [LRDP02]. Hence, battery power is a very limited resource for user devices in wireless networks and in fact, there have been several works in current literature [HSWC00, LH01, SCI⁺01] that concentrate only on making efficient use of battery power. Therefore, GKA protocols designed for wireless networks should use as little energy as possible in order to prolong the battery life in user devices. The GKA protocols designed for wireless networks also has to ensure that communication (transmission and reception of messages) costs are kept as low as possible because communication bandwidth in wireless networks is a limited resource and a GKA protocol that sends too many messages would result in a big drain of user devices’ batteries. In the following, we highlight four challenges for GKA in wireless networks.

1.3.1 Authentication

Authentication is a very important requirement for GKA in wireless networks. Without authentication, a malicious adversary can simply pretend to be a legitimate user node and join the GKA to obtain the shared group key. The adversary can then use the group key to listen in or send invalid messages to the group, thus resulting in the failure of secure group communications. The current approach [BCEP04, BC04, BCP07] to provide authentication in GKA protocols is to use digital signatures where the nodes taking part in GKA will sign and verify all the messages sent and received respectively. Before using the public keys of the sender to verify the digital signatures, the receivers will first have to receive and verify the certificates containing the valid public keys of the sender. Hence, depending on the group size and underlying GKA protocol, the network nodes may have to sign and verify a lot of messages, resulting in loss of efficiency. Furthermore, digital signatures schemes such as DSA [DSA94] consumes high amount of energy because of the use of energy-intensive computations such as modular exponentiation that studies [CKM00, HV02] have shown to have energy costs that are orders of magnitude *higher* than the energy costs of encryption, decryption and hash computations.

1.3.2 Dynamic Topology in Wireless Networks

As compared to wired network, certain types of wireless networks have higher dynamic topology due to more frequent dynamic group membership events such as User Join, User Leave and Network Merge. The reason for this highly dynamic topology is because user nodes in wireless networks are generally mobile and battery-operated PDAs and laptop computers as compared to user nodes in wired networks which are typically desktop systems with no mobility. A wireless node may move out of connectivity from the network and thus result in a user leave event. Similarly, if a wireless node runs out of batteries, then it would also trigger a user leave event. Whenever such dynamic group membership events such as user join, user leave and network merge occur, the GKA protocol has to ensure that the current group key is changed in order to provide forward secrecy (leaving users cannot obtain the new group key used) and backward secrecy (new users cannot obtain the old group key used).

1.3.3 Scalability of Wireless Networks

Some wireless networks such as WMNs and WSNs are large-scale wireless networks containing hundreds or thousands of nodes. The GKA protocol used in large-scale wireless network may require user nodes to perform many computations and send and/or receive many messages. As a result, the communication and computational costs of user nodes would be very high and the limited energy resource (batteries) of user nodes will be depleted more quickly.

1.3.4 Denial-of-Service Attacks

One important security requirement in secure group communications is the availability of GKA services to establish a group key without disruptions. However, GKA protocols are susceptible to Denial-of-Service (DoS) attacks that can disrupt GKA services and prevent the nodes in the network from establishing a common group key. In a DoS attack on GKA protocols, the malicious adversary may bring down GKA services by launching many invalid start GKA requests. This will cause honest nodes to repeatedly restart the GKA protocol, which consists of CPU-intensive cryptographic operations such as modular exponentiations, without establishing any group key. The adversary may also disrupt GKA services by repeatedly sending invalid messages during the protocol

run. For example, in a recent password-based GKA protocol, PAGKE [KJL06], an adversary can simply launch a DoS attack by sending invalid messages in Round 1 or Round 2 without knowing the password pw used. The honest user nodes will still perform energy-intensive modular exponentiations before discovering the invalid messages. The protocol will then halt and be re-started again. However, the adversary can continue to send invalid messages in every re-started GKA protocol run and cause the failure of GKA services as no final group key will be established.

1.4 Motivation

From the issues discussed in the previous section, it can be seen that the design of GKA protocols for wireless networks is more difficult than designing GKA protocols for wired networks due to the inherent characteristics of the wireless networks. Current literature of GKA protocols only discuss the complexity analysis [AKNR⁺04, KPT04a, AFP06, BCP07, Tse07], i.e. the number of modular exponentiations required and the number of messages sent and do not discuss about the energy consumption by group members, which is an important factor in wireless networks as wireless nodes are typically low power energy constrained devices that run on batteries. As mentioned earlier, there is also a need for GKA protocols to be energy-efficient in terms of both computational and communication costs. Furthermore, current complexity analysis of communication costs in GKA protocols only concentrate on the number of messages sent and not the number of messages received by a node. It is important to also analyze the energy costs of receiving messages in GKA protocols because energy is consumed when a node receives messages. For example, a GKA protocol may require each user to receive only 1 message, which would seem to be very efficient in terms of complexity analysis. However, that single received message could be a very large message containing information for all users in the group to compute the group key and hence each user node would still require a lot of energy to receive this message, which implies that the GKA protocol is not communication-efficient as opposed to the result shown in the complexity analysis [NLKW04, NLK⁺04, BLLH06].

1.4.1 Authentication

The current approach of using digital signature to provide authentication may not be the most efficient solution as user nodes have to first verify the certificates containing the public keys of the sender before using the public keys to verify the received signature. Hence if the group size is large and the receiver has to receive messages from every node, then the computational costs would be large. Hence, GKA protocols for wireless networks that uses digital signature for authentication should reduce the number of signature generation and verification. As the reception and verification of digital certificates also require a certain amount of energy, alternative methods for authentication such as ID-based or password-based solutions, both of which do not require the use of digital certificate, could be considered for GKA protocols in wireless networks.

1.4.2 Dynamic Nature of Wireless Networks

Some wireless networks especially mobile wireless networks have highly dynamic network topology as compared to wired networks. Hence, dynamic GKA protocols for wireless networks such as Join Protocol, Leave Protocol and Merge Protocol have to handle dynamic events such as user join, user leave and network merge efficiently by reducing the number of computations to be performed and the number of messages to be sent and received by each user node due to these dynamic events.

1.4.3 Scalability

Group Key Agreement protocols for large-scale wireless networks have to be scalable and energy efficient despite the large number of user nodes involved in the GKA protocol run. This is especially hard in contributory GKA as all nodes need to contribute to the creation of the final group key, hence the number of messages sent and received in the protocol run would also be large in a large-scale wireless network.

1.4.4 Denial-of-Service Attacks

Most GKA protocols in current literature do not consider protection against DoS attacks. However, the threat of DoS attacks is very real in actual implementation of GKA protocols in wireless networks because DoS attacks can be easily launch by the malicious adversary. Therefore, GKA

protocols for wireless networks have to provide protection against DoS attacks in order for the GKA protocol to be practical in real-world secure group communication applications.

1.5 Research Objectives

The main objective of this thesis is to study and design Group Key Management (in particular, contributory Group Key Agreement (GKA)) protocols for wireless networks that are more efficient (both in terms of complexity and energy costs) than current GKA protocols. GKA protocols in current literature will be studied and analyzed in detail to provide more in-depth understanding of these GKA protocols. In addition, we will also introduce energy analysis into GKA protocols to give a better understanding of the energy consumed by user nodes in wireless networks when executing GKA protocols. The characteristics and issues in different types of wireless networks will also be studied in detail and taken into consideration when designing the proposed GKA protocols so as to achieve GKA protocols that are more suitable for use in these wireless networks. We will focus on various types of wireless networks including Mobile Ad hoc Networks (MANETs) and large-scale wireless networks such as Wireless Mesh Networks (WMNs).

To achieve the main objective of this thesis, the following research goals that are important in the design of GKA protocols for wireless networks are defined:

1. To study current GKA protocols in order to gain better understanding and improve on efficiency in terms of authentication, computation and communications aspects.
2. To examine current dynamic GKA protocols, namely Join, Leave and Merge Protocol, and design more efficient dynamic GKA protocols for wireless networks.
3. To design scalable and energy-efficient GKA protocols for large-scale wireless networks.
4. To study the problem of Denial-of-Service (DoS) attacks in wireless networks and design GKA protocols that provide protection against DoS attacks.

1.6 Major Contribution of the Thesis

The contributions of this thesis fall into the following four subjects:

1. Identification and analysis of problems in authentication of GKA protocols in wireless networks, followed by solutions and evaluations.
2. Review of current efficient dynamic GKA protocols for wireless networks and development of even more efficient solutions.
3. Design of GKA protocols that are scalable and efficient for large-scale wireless networks.
4. Analysis of DoS attacks in GKA protocols, followed by solutions and evaluations.

The contributions in the four subjects mentioned above are as follows:

- Studied an efficient and popular GKA protocol for wireless network (the Burmester-Desmedt (BD) [BD95] protocol) in detail and developed more efficient authenticated GKA protocol based on the BD.
- Identified security flaws in current dynamic GKA protocols [STW98,DB05] and designed more efficient and secure dynamic GKA protocols for MANETs.
- Proposed efficient ID-based authenticated dynamic GKA protocols and provided energy analysis to show that the proposed ID-based protocols are not only efficient in terms of complexity analysis but also energy-efficient for user nodes in wireless networks.
- Developed several hierarchical group models/structures and proposed GKA protocols for these hierarchical group models/structures to achieve scalability and energy efficiency in large-scale symmetric and asymmetric wireless networks such as ad hoc networks and WMNs respectively.
- Identified three DoS attacks against a claimed DoS-resistant Two-party Anonymous Access Control (AAC) protocol [WDBA05] and proposed a better DoS-resistant Two-Party AAC protocol.
- Proposed DoS-resistant authenticated dynamic GKA protocols for wireless networks (including large-scale wireless networks) that not only provide protection against DoS attacks, but also are energy efficient and scalable.

The research in this thesis has been published in several journal articles and conference proceedings as shown in the Author's Publications list on page 285. This thesis provides the extended and improved versions of the published journal and conference papers.

1.7 Structure of the Thesis

The structure of this thesis is outlined as follows.

Chapter 2 In this chapter, we first cover the mathematical background and definitions used throughout this thesis. Next, we present the cryptographic tools used such as encryption schemes, digital signature schemes and hash functions. We then discuss various wireless networks and key establishment concepts briefly. Next, we review on the current literature of GKA protocols and present some attacks on GKA protocols. Lastly we evaluate on the current literature in DoS research.

Chapter 3 In this chapter, we examine on the computational and communication energy analysis used to analyze the energy costs of running GKA protocols for typical user nodes in wireless networks. We first examine the computational energy costs of two variants of the SA-1110 "StrongARM" microprocessor, namely the 133MHz and 206MHz "StrongARM" microprocessors. The 133MHz "StrongARM" microprocessor is commonly used in some PDAs and other user handheld devices and also in sensor nodes such as Rockwell Scientific WINS sensor nodes while the 206MHz "StrongARM" microprocessor is commonly used in PDAs such as the Compaq iPAQ H3670 PDA. Next, we consider the communication energy analysis by examining three wireless communication devices, namely the 100kbps radio transceiver module used on the Rockwell Scientific WINS nodes, the IEEE 802.11 Spectrum24 LA-4121 WLAN card and Lucent IEEE 802.11 11 Mbps WaveLAN card that are both commonly used by PDAs and laptop computers for wireless communications.

Chapter 4 In this chapter, we explore the authentication issue in GKA for wireless networks. First, we identify an interesting characteristic of a popular and efficient GKA protocol that is suitable for wireless networks, namely the Burmester-Desmedt (BD) GKA protocol. Next, we design an efficient authenticated GKA protocol for wireless networks using this characteristic of

the BD. We compare our proposed protocol with current methods to authenticate the BD and show that our proposed protocol is more efficient.

Chapter 5 In this chapter, we discuss the issue of highly dynamic topology in MANETs, which is a type of symmetric wireless network. We first show a security flaw in the method briefly mentioned by Steiner et al [STW98] to provide a Join protocol for the BD protocol. Next we show another security flaw in the provable-secure Leave protocol of Dutta and Barua [DB05]. We then proposed four authenticated GKA protocols namely Initial GKA, Join, Leave and Merge protocols for MANETs. Our proposed protocols provide autoconfiguration. The complexity analysis of our proposed protocols with four dynamic GKA protocols shows that our protocols are efficient and suitable for MANETs.

Chapter 6 In this chapter, we explore ID-based authentication which is another alternative to digital signature authentication. Besides authentication, we also deal with dynamic (Join, Leave and Merge) issues and propose an ID-based authenticated Initial GKA protocol and three ID-based authenticated dynamic GKA protocols, namely Join, Leave and Merge Protocol for symmetric wireless networks. The use of ID-based authentication in our proposed protocol is more efficient than digital signatures because ID-based authentication do not require digital certificates [GQ88]. Besides complexity analysis, we also provide an energy analysis that shows that our proposed protocols are more efficient and the energy costs for running our proposed protocols are significantly lower. The complexity and energy analysis are based on the 133MHz “StrongARM” microprocessor and two transceivers commonly used in wireless networks [CKM00].

Chapter 7 In this chapter, we focus on the scalability problem in static large-scale symmetric ad hoc networks. First, we developed a hierarchial group structure known as the Generalized Circular Hierarchical (C-H) Group Model, then we design a GKA protocol for the C-H group model. The complexity and energy analysis of the proposed protocol shows that it is more scalable and energy-efficient as compared to current GKA protocols for large-scale symmetric ad hoc networks.

Chapter 8 In this chapter, we focus on static symmetric heterogeneous wireless networks which contains both high power nodes that have little energy constraints and low power energy constrained

user nodes. We first design a Contributory Ring-Centralized Group Model for heterogeneous wireless networks and then propose a GKA protocol to improve the scalability and energy efficiency of nodes in heterogeneous wireless networks. The complexity and energy analysis of our proposed protocol indicates that our proposed protocol is more efficient and suitable than current GKA protocols for use in static heterogeneous wireless networks.

Chapter 9 In this chapter, besides focusing on scalability and energy efficiency issues in dynamic asymmetric heterogeneous wireless networks, we also focus on the important problem of DoS attacks on GKA protocols. First, we developed the Centralized-Contributory Ring Group Model which provides scalability and energy efficiency. Next, we proposed the authenticated initial GKA protocol and dynamic GKA protocols, namely Join and Leave Protocol, that provides protection against DoS attacks. We also developed a method to detect malicious group insider nodes that tries to disrupt the GKA protocol. Finally, we show with the complexity and energy analysis that our proposed protocols are more energy-efficient and scalable for use in heterogeneous wireless networks despite providing the important feature of protection against DoS attacks.

Chapter 10 In this chapter, we focus on password-based authentication techniques instead of digital signature authentication. First, we point out possible DoS attacks against a claimed DoS-resistant Anonymous Access Control (AAC) protocol [WDBA05]. We then proposed a new and improved DoS-resistant AAC protocol. Next, we continue to explore DoS attacks against GKA protocols, including password-based GKA protocols, in dynamic asymmetric wireless networks such as IEEE 802.11 Wireless LAN (WLAN) and WMNs. We proposed an authenticated Initial Password-based GKA protocol and two authenticated dynamic Password-based GKA protocols, namely the Join and Leave Protocol, that are energy-efficient, scalable and also provide protection against DoS attacks in wireless networks. The complexity and energy analysis indicates that our proposed Password-based GKA protocols are more efficient and scalable than current GKA protocols.

Chapter 11 In this chapter, we provide a summary of this thesis and present possible future research directions.

Chapter 2

Preliminaries

In this chapter, we cover the mathematical background and definitions necessary for understanding the remainder of this thesis. Necessary background materials on cryptographic tools, wireless networks and key establishment concepts are presented. We also present current literature of GKA and show some attacks on GKA protocols. Lastly, we present the current literature in DoS research.

2.1 Mathematical Background

In this section, we introduce the basic ideas of complexity theory, introduce different cryptographic definitions and provide the necessary mathematical background required in this thesis.

2.1.1 Abstract Algebra and the Main Groups

Abstract algebra is the field of mathematics that studies algebraic structures. In cryptography, most computations are done in algebraic structures such as finite groups, rings and fields [Sti05]. An algebraic structure is defined as the collection of a set with one or more operations which can be performed on elements of the set [Sti05]. The following gives an introduction of finite cyclic groups which is used in this thesis.

Finite cyclic group G : A group is an algebraic structure consisting of a set G of group elements and a binary group operation $*$: $G \times G \rightarrow G$ such that the following conditions hold:

1. If $a, b \in G$, then $a \cdot b \in G$
2. The group operation is associative. That is, $a \cdot (b \cdot c) = (a \cdot b) \cdot c \forall a, b, c \in G$
3. There is an identity element $1 \in G$ such that $a \cdot 1 = 1 \cdot a \forall a \in G$
4. For each $a \in G$, there is an inverse $a^{-1} \in G$ such that $a \cdot a^{-1} = a^{-1} \cdot a = 1 \cdot a$

The group order of G is the number of elements in G and is denoted as $|G|$. A group G is finite if $|G|$ is finite. G is said to be cyclic if there is an element $g \in G$ such that for each $a \in G$, there is an integer i with $a = g^i$. Such an element g is called the generator of G . Below, we list some of the cyclic groups that are used throughout the thesis.

Z_p^* : The multiplicative groups of integers modulo a prime p with group order $\Phi(p) = p - 1$ having at least one large prime factor, where Φ is Euler's totient function. In this thesis, we use the subgroup of Z_p^* of prime order q , where q is a large prime factor that divides $p - 1$ (q is usually 160-bits while p is usually 1024-bits).

Z_n^* : The multiplicative groups of integers modulo a product n of two (or more) large strong primes p and q with $p - 1$ and $q - 1$ containing at least one large prime factor. The factoring of n is usually assumed to be hard.

2.1.2 Elliptic Curve Cryptography (ECC)

Elliptic curve cryptography (ECC) is an approach to public-key cryptography based on the algebraic structure of elliptic curves over finite fields. The use of elliptic curves in cryptography was suggested independently by Neal Koblitz [Kob87] and Victor S. Miller [Mil86] in 1985. In this section, we shall briefly introduce ECC and we refer readers to books [Men93] and [KMWZ98] for further information on ECC.

An elliptic curve is defined in the following equation where:

$$y^2 + cxy = x^3 + ax + b$$

An example of elliptic curve taken from [Sti05] is given in Figure 2.1.

Figure 2.1: Elliptic curve $y^2 = x^3 - 8x + 6$

If the coordinates x and y are chosen from a large finite field, the solutions form a finite abelian group. The discrete logarithm problem (see Section 2.2.3) on such elliptic curve groups is believed to be more difficult than the corresponding problem in (the multiplicative group of nonzero elements of) the underlying finite field. Thus keys in ECC can be chosen to be much shorter for a comparable level of security. Elliptic curves used in cryptography are typically defined over two types of finite fields: 1) F_p (where $p \geq 3$ is a large prime number) and 2) F_{2^m} (a binary representation with 2^m elements).

Elliptic Curve Groups over F_p

F_p is the field of integers modulo p and consists of all the integers from 0 to $p - 1$. An elliptic curve with the underlying field of F_p can be formed by choosing the variables a and b within the field of F_p . The elliptic curve includes all points (x, y) which satisfy the elliptic curve equation modulo p (where x and y are numbers in F_p). The elliptic curve equation is defined as

$$y^2 = x^3 + ax + b \mod p$$

where $a, b \in F_p$ are constants. If $4a^3 + 27b^2 \neq 0$, then the elliptic curve can be used to form a

group. An elliptic curve group over F_p consists of the points on the corresponding elliptic curve, together with a special point O called the point at infinity. The group that consist of all the points on such an elliptic curve, together with the point at O is denoted as $E(F_p)$ and the order of the group is denoted as $|E(F_p)|$.

Elliptic Curve Groups over F_{2^m}

Elements of the field F_{2^m} are m-bit strings. The rules for arithmetic in F_{2^m} can be defined by either polynomial representation or by optimal normal basis representation. Since F_{2^m} operates on bit strings, computers can perform arithmetic in this field very efficiently. As a result of the field F_{2^m} having a characteristic 2, the elliptic curve equation is slightly adjusted for binary representation:

$$y^2 + xy = x^3 + ax^2 + b$$

where $a, b \in F_{2^m}$ are constants and $b \neq 0$. The elliptic curve includes all points (x, y) which satisfy the elliptic curve equation over F_{2^m} (where x and y are elements of F_{2^m}). An elliptic curve group over F_{2^m} consists of the points on the corresponding elliptic curve, together with a point at infinity O . The group that consist of all the points on such an elliptic curve, together with the point at O is denoted as $E(F_{2^m})$ and the order of the group is denoted as $|E(F_{2^m})|$.

Point Addition

Two points P and Q with different coordinates on the elliptic curve can be added together as shown in Figure 2.1. The addition of points is described as follows:

- i Construct a line which intersects both P and Q . This line meets the curve in the third point R as shown Figure 2.1.
- ii The inverse of the point R is the required element for $P + Q$.

Point Multiplication

Point multiplication is the dominant operation in ECC cryptographic schemes. Point multiplication is defined as calculating $Q = kP$ where k is an integer and P is a point on the elliptic curve. The

inverse operation to point multiplication is the elliptic curve discrete logarithm problem which is defined as : Give Q and P , find integer k such that $Q = kP$.

- i To compute point multiplication of kP , the point P is added to itself k times.
- ii On the elliptic curve, the point P is added to itself by constructing a tangent line to the curve at the point P . This tangent line will meet the curve at the third point R' . The inverse of the point R' will be the required element for $P + P$.

2.1.3 Bilinear Maps from Elliptic Curve Pairings

A bilinear map is defined as $e : G_1 \times G_1 \rightarrow G_2$ where G_1 is an additive cyclic group of prime order q , G_2 is a multiplicative cyclic group of the same order and P is an arbitrary generator of G_1 [BF03].

An admissible bilinear map has the following properties:

Bilinear: For every $R, S, T \in G_1$ and $a, b \in \mathbb{Z}_q^*$, $e(aR, bS) = e(R, S)^{ab}$. This can be rewritten as $e(R + S, T) = e(R, T)e(S, T)$ and $e(R, S + T) = e(R, S)e(R, T)$.

Non-degenerate: There exists $R, S \in G_1$ such that $e(R, S) \neq I_{G_2}$ where I_{G_2} denotes the identity element of the group G_2 .

Computable: There exist an efficient algorithm to compute $e(R, S) \forall R, S \in G_1$.

2.2 Computational Problems

We present here several computational problems and assumptions such as the Integer factoring problem, the RSA problem, the Discrete Logarithm problem and the Computational Diffie-Hellman problem as defined in [MOV97]. The security of many public-key cryptosystems are based on the intractability of these problems. Similarly, the security of some of the proposed Group Key Management (GKM) protocols in this thesis are also based on the intractability of these problems.

2.2.1 Integer Factoring Problem

The intractability of the integer factoring problem forms the basis for the security of public-key cryptosystem such as RSA public-key encryption scheme, RSA signature scheme and the Rabin public-key encryption scheme. The integer factoring problem is defined as follows: Given a positive

integer n , find its prime factorization, that is, write $n = p_1^{e_1} p_2^{e_2} \cdots p_m^{e_m}$ where p_i are pairwise distinct primes and each $e_i \geq 1$ for $1 \leq i \leq m$.

2.2.2 RSA Problem

The security of the RSA signature scheme [RSA78] and the RSA public-key encryption scheme [RSA78] are based on the intractability of the RSA problem. The RSA problem is based on the integer factorization problem as described above. The RSA problem is defined as follows: Given a positive integer n that is the product of two distinct odd primes p and q , a positive integer e such that the greatest common divisor (gcd) of e and $(p-1)(q-1)$ is 1, i.e. $\gcd(e, (p-1)(q-1)) = 1$, and an integer c , find an integer m such that $m^e = c \pmod{n}$.

2.2.3 Discrete Logarithm Problem

The security of cryptographic techniques such as Diffie-Hellman [DH76] key agreement, ElGamal encryption, ElGamal signature scheme are dependent on the intractability of the discrete logarithm problem. The Discrete Logarithm Problem is defined as follows: Given a prime p , a generator g of Z_p^* and an element $b \in Z_p^*$, find the integer a such that $g^a = b \pmod{p}$.

2.2.4 Computational Diffie-Hellman CDH Problem

The Computational Diffie-Hellman (CDH) problem is closely related to the Discrete Logarithm Problem [BL96] as defined above. The CDH problem is defined as follows: Let $G = \langle g \rangle$ be a cyclic group of prime order q and an element g be the generator of G . Given the group G , and two elements g^a and $g^b \in G$ for randoms a and $b \in Z_q^*$, compute g^{ab} . It is assumed that the CDH problem is difficult. That is, there is no polynomial time algorithm A that can solve the CDH problem with non-negligible probability ϵ .

$$|Pr[a, b \leftarrow Z_q^* : A(g, g^a, g^b) = g^{ab}]| \geq \epsilon$$

2.2.5 Decisional Diffie-Hellman DDH Problem

The Decisional Diffie-Hellman (DDH) problem [DH76] is defined as follows: Let $G = \langle g \rangle$ be a cyclic group of prime order q and an element g be the generator of G . Given the group G , and two elements g^a and $g^b \in G$ for randoms a and $b \in \mathbb{Z}_q^*$, the following two tuples of random variables $(g^a, g^b, g^{ab} \bmod p)$ and (g^a, g^b, R) , where R is a random value in G , are computationally indistinguishable. That is, there is no polynomial time algorithm $\bar{A}$ that can solve the DDH problem with non-negligible probability ϵ' .

$$|Pr[\bar{A}(g^a, g^b, g^{ab}) = true] - Pr[\bar{A}(g^a, g^b, R) = true]| \geq \epsilon'$$

2.3 Cryptographic Concept

In this section, we discuss the various properties that may be provided by cryptographic algorithms so as to give the reader a better understanding of the cryptographic methods and algorithms used throughout this thesis. Cryptography is defined as the study of mathematical techniques related to aspects of information security such as confidentiality, data integrity, entity authentication and data origin authentication [MOV97]. The four fundamental goals in information security that may be achieved by cryptographic algorithms are defined in [MOV97] and [Sti05] as follows:

1. **Confidentiality** ensures that information flow between the sender and receiver is unintelligible to outsiders (i.e. data is only available to authorized recipients). In cryptographic protocols, confidentiality is important to ensure that keys and other data are available only as intended.
2. **Integrity** means that data has not been altered by unauthorized entities. Data Integrity is important in most cryptographic protocols to protect elements such as identity fields and nonces.
3. **Authentication** is a process of identifying either the entity or the data itself. Entity authentication allows two (or more) parties entering a communication to identify each other. Data origin authentication, which is a fundamental step in achieving entity authentication in

protocols including key establishment protocols, guarantees the origin of the data delivered over a channel. It should be noted that data origin authentication implies data integrity since altering the data must have altered its origin.

4. **Non-repudiation** ensures that senders cannot deny sending data that they have committed to. This is typically provided using a digital signature scheme.

Next we discuss cryptographic algorithms that are used to achieve the above-mentioned cryptographic goals:

2.3.1 Message Authentication Code

A message authentication code scheme MACS [MOV97] is a key-based algorithm that associates an authentication tag with every valid message. The authentication tag for a message can be efficiently verified by the other party sharing the same key used to create the tag. An adversary that sees many message-tag pairs is not able to forge a tag on new messages. The definition of a message authentication code scheme MACS is as follows:

Definition 1 *A message authentication code scheme MACS is given by three algorithms where:*

- $\mathcal{K}$, the key generation algorithm, takes a security parameter l and returns a key k .
- MAC_k , the tagging or message authentication algorithm, which takes as inputs key k and message $m \in \{0,1\}^*$ and returns an authentication tag μ defined as $\mu = \text{MAC}_k(m)$.
- VER_k , the verification algorithm which takes as inputs a key k , message $m \in \{0,1\}^*$ and authentication tag μ and returns $\text{VER}_k(m, \mu) = \text{Valid}$ for "accept" and $\text{VER}_k(m, \mu) = \text{Invalid}$ for "reject" of the authentication tag.

We require that for all k which can be output by $\mathcal{K}(l)$, for all $m \in \{0,1\}^*$, and for all $\mu = \text{MAC}_k(m)$, we have $\text{VER}_k(m, \mu) = \text{Valid}$. We also require that $\mathcal{K}$, MAC_k and VER_k can be computed in polynomial time.

There are three types of attacks, namely known text attack, chosen-message attack and adaptive chosen-message attack, against MAC schemes [MOV97]. In this thesis, we shall only consider the

security of a MACS against adaptive chosen-message attack, the strongest of the three attack. A MACS is secure against adaptive chosen-message attack if for any probabilistic polynomial time forging algorithms $\mathcal{F}$, the following is negligible (in l):

$$Pr[k \leftarrow \mathcal{K}(l); (m, \mu) \leftarrow \mathcal{F}^{\text{MAC}_k(\cdot)}(l) : \text{VER}_k(m, \mu) = \text{Valid}]$$

where in an adaptive chosen-message attack, the adversary $\mathcal{A}$ is allowed to ask the MAC generator to generate authentication tags for a number of messages of $\mathcal{A}$'s choice. The choice of these messages may depend on previously obtained authentication tags.

Examples of MAC function include Hashed MAC (HMAC) [BCK96, KBC97] and Cipher-Block-Chaining MAC (CBC-MAC) [BKR94].

2.3.2 Cryptographic Hash Function

A cryptographic hash function $H : \{0, 1\}^* \rightarrow \{0, 1\}^\lambda$ maps variable messages in $\{0, 1\}^*$ to a finite set of λ -bit hash values. A cryptographic hash function H is secure if it satisfies the following properties:

- *Preimage resistant:* Given H and hash value $H(m)$, it is computationally hard to compute the preimage m .
- *Collision resistant:* Given H , it is computationally hard to find two messages m and m' such that $m \neq m'$ and $H(m) = H(m')$.
- *Second preimage resistant:* Given H and message m , it is computationally hard to find a message m' such that $m' \neq m$ and $H(m') = H(m)$.

Examples of popular hash functions includes Secure Hash (SHA) [oS95, ErJ01], Message Digest 5 (MD5) [Riv92] and RIPEMD-160 [DBP96].

2.3.3 Encryption Scheme

A cryptographic solution to achieve confidentiality is through the use of encryption. That is, encryption allows one party to send a message to another such that the contents of the message

remains hidden from anyone listening to the communication. An adversary $\mathcal{A}$ should not be able to learn a plaintext information from its ciphertext.

The definition of an encryption scheme is as follows:

Definition 2 *An encryption scheme consists of three algorithms as follows:*

- $\mathcal{K}$, the key generation algorithm that takes a security parameter l as input and outputs a valid encryption key K and valid decryption key K^{-1} .
- $\mathcal{E}$, the encryption algorithm, which takes the encryption key K and message $m \in \{0,1\}^*$ as inputs and outputs ciphertext $c = \mathcal{E}_K(m)$.
- $\mathcal{D}$, the decryption algorithm that takes the decryption key K^{-1} and ciphertext c as inputs and outputs message $m = \mathcal{D}_{K^{-1}}(c)$.

We require that for all (K, K^{-1}) which can be output by $\mathcal{K}(l)$, for all $m \in \{0,1\}^*$, and for all $c = \mathcal{E}_K(m)$, we have $\mathcal{D}_{K^{-1}}(c) = m$. We also require that $\mathcal{K}$, $\mathcal{E}$ and $\mathcal{D}$ can be computed in polynomial time.

An encryption scheme is known as a *symmetric* key encryption scheme SKE if $K = K^{-1}$. In an *asymmetric* or public key encryption scheme PKE, K and K^{-1} are different and it is computationally hard to obtain the private key K^{-1} from the public key K .

Indistinguishability of encryptions (IND) is an important security property of encryption schemes (both symmetric key encryption scheme SKE and asymmetric or public key encryption scheme PKE). The property of indistinguishability under chosen-plaintext attack (IND-CPA) is considered a basic requirement for encryption schemes. IND-CPA is equivalent to the property of semantic security [GM84]. Besides IND-CPA, two other definitions for indistinguishability are as follows: indistinguishability under (non-adaptive) chosen ciphertext attack (IND-CCA), and indistinguishability under adaptive chosen ciphertext attack (IND-CCA2). An encryption scheme which is IND-CCA secure is also IND-CPA secure, and a scheme which is IND-CCA2 secure is both IND-CCA and IND-CPA secure. Thus, IND-CCA2 is the strongest of the these three definitions of security. We shall discuss security under IND-CCA2 in this thesis.

For an encryption scheme, indistinguishability under adaptive chosen ciphertext attack (IND-CCA2) is defined by the following game between an adversary $\mathcal{A}$ and a challenger. The game is defined as follows:

1. A challenger generates (K, K^{-1}) by running the key generation algorithm $\mathcal{K}$.
2. The adversary $\mathcal{A}$ may perform any number of encryptions and queries to the decryption oracle based on arbitrary ciphertexts. Finally, the adversary submits two distinct chosen plaintexts m_0, m_1 to the challenger.
3. The challenger selects a bit $b \in \{0, 1\}$ at random, and sends the "challenge" ciphertext $c^* = \mathcal{E}_K(m_b)$ back to $\mathcal{A}$.
4. $\mathcal{A}$ will make a number of queries to the encryption oracle and only in IND-CCA2, $\mathcal{A}$ will also make further queries to the decryption oracle, but may not submit the challenge ciphertext c^* for query to the decryption oracle.
5. Finally, $\mathcal{A}$ outputs a guess b' for b .

The adversary wins the game if $b' = b$. Hence, an encryption scheme (both SKE and PKE) is secure against IND-CCA2 if for any probabilistic, polynomial time adversary $\mathcal{A}$, winning the above game is negligible, i.e. the following advantage is negligible (in l):

$$\text{Adv}_{\text{SKE/PKE}, \mathcal{A}}(l) = 2 \times |\Pr[b' = b]| - 1$$

Examples of symmetric key encryption scheme include DES [Nat93], RC5 [Riv95], IDEA [LM90], Blowfish [Sch94] and AES [dae02] which is based on the Rijndael cipher [DR00]. Examples of asymmetric key encryption scheme include RSA [RSA78] and Elliptic Curve Cryptography (ECC) [Men93].

2.3.4 Digital Signature Scheme

Digital signature schemes [MOV97] provide the authentication, data integrity and non-repudiation services. The definition of a digital signature scheme is as follows:

Definition 3 A digital signature scheme is given by three algorithms as follows:

- $\mathcal{K}$, the key generation algorithm that takes a security parameter l as input and returns a key pair (sk, vk) , where sk is the signing key and vk is the verification key.
- $\mathcal{S}$, the signature algorithm, which takes the signing key sk and message $m \in \{0, 1\}^*$ as inputs and outputs a signature $\sigma = \mathcal{S}_{sk}(m)$.
- $\mathcal{V}$, the verification algorithm which takes the verification key vk , message $m \in \{0, 1\}^*$ and signature σ as inputs and outputs $\mathcal{V}_{vk}(m, \sigma) = \text{Valid}$ if the signature is valid or $\mathcal{V}_{vk}(m, \sigma) = \text{Invalid}$ if the signature is invalid.

We require that for all (sk, vk) which can be output by $\mathcal{K}(l)$, for all $m \in \{0, 1\}^*$, and for all $\sigma = \mathcal{S}_{sk}(m)$, we have $\mathcal{V}_{vk}(m, \sigma) = \text{Valid}$. We also require that $\mathcal{K}$, $\mathcal{S}$ and $\mathcal{V}$ can be computed in polynomial time.

A digital signature scheme is secure under adaptive chosen message attack if for any probabilistic polynomial time forging algorithms $\bar{\mathcal{F}}$, the following is negligible (in l):

$$Pr[(sk, vk) \leftarrow \mathcal{K}(l); (m, \sigma) \leftarrow \bar{\mathcal{F}}^{\mathcal{S}_{sk}(\cdot)}(l, vk) : \mathcal{V}_{vk}(m, \sigma) = \text{Valid}]$$

where in an adaptive chosen-message attack, the adversary $\mathcal{A}$ is allowed to ask the signer to sign a number of messages of $\mathcal{A}$'s choice. The choice of these messages may depend on previously obtained signatures.

Below we list some digital signature schemes that are popular and also used in this thesis:

Schnorr Signature Scheme

Schnorr [Sch91] designed a computationally-efficient signature scheme in 1991 that is suitable for use in smart cards with limited computational power. Besides smart cards, Schnorr's signature scheme is also suitable for devices in wireless networks due to its efficiency. The signature scheme is described as follows:

Key Generation : Select two random large primes, q and p , such that q divides $p - 1$ (p is normally 1024 bits and q is normally 160 bits). Next, pick an element $g \in \mathbb{Z}_p^*$ of order q . Choose a random

$x \in Z_q^*$ and compute $y = g^x \bmod p$. Let H be a cryptographic hash function where $H : \{0, 1\}^* \rightarrow Z_q$. Then, the resulting public key is (p, q, g, y, H) and the private key is x .

Signature Generation : To sign a message $m \in \{0, 1\}^*$ using the private key x , perform the following:

First, the signer choose a random $k \in Z_q^*$ and compute $\tau = g^k \bmod p$. Set $c = H(m || \tau)$ and compute $s = k + cx \bmod q$. Finally output the signature pair (s, c) as the signature on m .

Signature Verification : To verify a message/signature pair $(m, (s, c))$ using the public key (p, q, g, y, H) , the verifier computes $v = g^s y^{-c} \bmod p$ and accepts the signature if $c = H(m || v)$. Otherwise, reject the signature.

In fact, the Schnorr scheme is proven to be secure under the random oracle model [BR93], that is, the Schnorr scheme is secure against forgery.

Digital Signature Algorithm (DSA)

In 1994, a digital signature standard DSA [DSA94] was proposed which is based on the Elgamal signature scheme [Gam85]. The DSA scheme is described as follows:

Key Generation : Select two random large primes, q and p , such that q divides $p - 1$ (p is normally 1024 bits and q is normally 160 bits). Next, pick an element $g \in Z_p^*$ of order q . Choose a random $x \in Z_q^*$ and compute $y = g^x \bmod p$. Let H be a cryptographic hash function where $H : \{0, 1\}^* \rightarrow Z_q$. Then, the resulting public key is (p, q, g, y, H) and the private key is x .

Signature Generation : To sign a message $m \in \{0, 1\}^*$ using the private key x , perform the following:

First, the signer choose a random $k \in Z_q^*$, compute the inverse $(k^{-1} \bmod q)$ of $k \bmod q$ and compute $r = (g^k \bmod p) \bmod q$. Next, compute $s = k^{-1} \cdot (H(m) + x \cdot r) \bmod q$. Finally output the signature pair (r, s) as the signature on m .

Signature Verification : To verify a message/signature pair $(m, (r, s))$, the verifier computes the inverse $(s^{-1} \bmod q)$ of $s \bmod q$, $u = s^{-1} \cdot H(m) \bmod q$ and $v = s^{-1} \cdot r \bmod q$. Next, check $r = (g^u \cdot y^v \bmod p) \bmod q$. If they are equal, accepts the signature. Otherwise, reject the signature.

Elliptic Curve DSA (ECDSA)

In Section 2.1.2, we have discussed about Elliptic Curve Cryptography (ECC). In this section, we described the ECC variant of DSA, namely the Elliptic Curve DSA (ECDSA) scheme [JM99]. In

general, the security level of an ECDSA public key of 163-bit is equivalent to the security level of a DSA public key of 1024-bit. On the other hand, the signature size for both DSA and ECDSA are the same. The ECDSA is described as follows:

Key Generation : Initially, a cyclic group G of prime order q is selected and a generator P of G is chosen. Next, select a random $d \in Z_q^*$ and compute $Q = d \cdot P$. Let H be a cryptographic hash function where $H : \{0, 1\}^* \rightarrow Z_q$. Then, the resulting public key is Q and the private key is d .

Signature Generation : To sign a message $m \in \{0, 1\}^*$ using the private key d , perform the following:

First, the signer choose a random $k \in Z_q^*$, compute the inverse (k^{-1}) of k and compute $k \cdot P = (x_1, y_1)$ and $r = x_1 \bmod q$. Next, compute $s = k^{-1} \cdot (H(m) + d \cdot r) \bmod q$. Finally output the signature pair (r, s) as the signature on m .

Signature Verification : To verify a message/signature pair $(m, (r, s))$, the verifier computes the inverse (s^{-1}) of s , $u = s^{-1} \cdot H(m) \bmod q$ and $v = s^{-1} \cdot r \bmod q$. Next, compute $(\alpha_1, \beta_1) = u \cdot P + v \cdot Q$. Accept the signature if $\alpha_1 = r \bmod q$ is valid. Otherwise, reject the signature.

RSA Signature Scheme

The RSA signature scheme [RSA78] is a well-known signature scheme which is based on the factorization problem. The RSA signature scheme is described as follows:

Key Generation : Select two random 512 bits primes $\bar{p}$ and $\bar{q}$ and compute $\bar{N} = \bar{p}\bar{q}$ and $\Phi = (\bar{p} - 1)(\bar{q} - 1)$, where Φ is Euler's totient function. Next, select an integer $e = 2^{16} + 1 = 65537$ and compute secret exponent d such that $ed = 1 \bmod \Phi$, then the public key is $(\bar{N}, e)$ and the private key is $(d, \bar{p}, \bar{q})$.

Signature Generation : To sign message m , the signer computes the RSA signature $\sigma = H(m)^d \bmod \bar{N}$, where H is a cryptographic hash function.

Signature Verification : To verify a message/signature pair (m, σ) , the verifier computes $v = \sigma^e \bmod \bar{N}$ and accepts the signature if $H(m) = v$. Otherwise, reject the signature.

Timing Cost Comparison between RSA and DSA Signature scheme

Table 2.1 shows the results of simulating the RSA and DSA signature generation and verification using MIRACL library [Mir] on a 3GHz Pentium IV system. From the table, RSA signature gen-

Table 2.1: Timing Costs for RSA and DSA signature scheme on a 3GHz Pentium IV system

Operations	Time taken
1024-bit RSA Signature Generation	3.88ms
1024-bit RSA Signature Verification, $e = 65537$	0.22ms
1024-bit DSA Signature Generation (160-bit exponent)	2.30ms
1024-bit DSA Signature Verification (160-bit exponent)	2.72ms

eration and verification takes 3.88ms and 0.22ms respectively while the DSA signature generation and verification takes 2.30ms and 2.72ms respectively. Hence, the RSA signature verification is very fast, even for 1000 signatures verifications, it only takes 0.22 seconds. In comparison, it takes 2.72 seconds to verify 1000 signatures using the DSA signature scheme, which is around 12 times slower than RSA signature scheme.

Guillou and Quisquater ID-based Signature Scheme

Guillou and Quisquater [GQ88] designed an efficient GQ ID-based Signature scheme in 1990. The GQ signature scheme is described as follows:

Setup : The Private Key Generator (PKG) selects two random 512 bits primes $\bar{p}$ and $\bar{q}$ and compute $\bar{N} = \bar{p}\bar{q}$ and $\Phi = (\bar{p} - 1)(\bar{q} - 1)$, where Φ is Euler's totient function. Next, The PKG selects an integer $e = 2^{16} + 1 = 65537$ and compute secret exponent d such that $ed = 1 \pmod{\Phi}$. The PKG also selects a one way cryptographic hash function $H : \{0, 1\}^* \rightarrow \{0, 1\}^\lambda$, where λ is an integer. The parameters **params** are $(\bar{N}, e, H)$ and the master keys are $(\bar{p}, \bar{q}, d)$.

Extract : The PKG verifies the given user identity ID and computes the secret key for the identity as $S_{ID} = H(ID)^d \pmod{\bar{N}}$. The secret key S_{ID} is then sent securely to user ID .

Sign : Given a private key S_{ID} and a message m , choose $\tau \in Z_{\bar{N}}^*$ and compute $c = H(\tau^e, m)$ and $s = \tau \cdot S_{ID}^c \pmod{\bar{N}}$. Then $\sigma = (s, c)$ is the signature of m .

Verify : The signature $\sigma = (s, c)$ of an identity ID on a message m is valid if the equation $c = H(s^e \cdot (H(ID))^{-c}, m)$ holds.

Sakai-Ohgishi-Kasahara's ID-based Signature Scheme

Sakai et al [SOK00] designed an ID-based signature scheme known as SOK signature scheme using bilinear maps from elliptic curve pairings (as described in Section 2.1.3) in 2000. The SOK signature

scheme uses a bilinear map $e : G_1 \times G_1 \rightarrow G_2$ where G_1 and G_2 are cyclic group of the same order q and P be a generator of G_1 . The SOK signature scheme is as follows:

Setup : The Private Key Generator (PKG) chooses $s \in Z_q^*$ as his master secret key and computes the global public key $P_{pub} = sP$. It then chooses two Map-to-Point hash functions $H_1 : \{0, 1\}^n \rightarrow G_1$ and $H_2 : \{0, 1\}^* \rightarrow G_1$. The parameters **params** are $(G_1, G_2, e, P, P_{pub}, H_1, H_2)$ and the master key is s .

Extract : The PKG verifies the given identity ID and computes the secret key for the identity as $S_{ID} = sQ_{ID}$, where $Q_{ID} = H_1(ID)$.

Sign : Given a private key S_{ID} and a message m , choose a $t \in Z_q^*$ and compute $S_1 = S_{ID} + tH_2(m)$ and $S_2 = tP$. Then $\sigma = (S_1, S_2) \in G_1 \times G_1$ is the signature of message m .

Verify : The signature $\sigma = (S_1, S_2)$ of an identity ID on a message m is valid if the following equation holds.

$$e(Q_{ID}, P_{pub})e(H_2(m), S_2) = e(S_1, P) \quad (2.1)$$

Rabin's Signature Scheme

Rabin [Rab79] proposed an efficient Rabin's signature scheme which only requires each user to perform one modular multiplication per signature verification as compared to other signature schemes such as the ElGamal signature scheme [Gam85] that requires each user to perform modular exponentiations per signature verification. It should be noted that modular exponentiations are more computationally expensive than modular multiplication.

Setup : Let H be a one-way hash function. A signer U first generate $\bar{N} = \bar{p} \cdot \bar{q}$ where $\bar{p}$ and $\bar{q}$ are primes of 512 bits each such that $\bar{p} \equiv 3 \pmod{4}$ and $\bar{q} \equiv 3 \pmod{4}$. The public and private key of the signer U is $\bar{N}$ and $(\bar{p}, \bar{q})$ respectively.

Sign : To sign a message m , signer U first randomly choose τ and compute $a = H(m, \tau)$. If $a^{\frac{\bar{p}-1}{2}} \not\equiv 1 \pmod{\bar{p}}$ or $a^{\frac{\bar{q}-1}{2}} \not\equiv 1 \pmod{\bar{q}}$, then the signer U randomly chooses another τ and repeat the signing process. Otherwise, U computes the four modular square roots $(\sigma_1, \sigma_2, \sigma_3, \sigma_4)$ of $\sigma_i^2 = a \pmod{\bar{N}}$ and chooses one of σ_i for $i \in \{1, 2, 3, 4\}$ as σ , i.e. $\sigma = \sigma_i$. The signature of message m is (σ, τ) .

Verify : To verify a signature (σ, τ) of message m , the verifier checks whether $\sigma^2 = H(m, \tau) \pmod{\bar{N}}$. If the equation is valid, it means that the signature on m is valid. Otherwise, reject the signature.

2.4 Wireless Networks

Security and authentication are more important in wireless communication than in wired communication because of the open nature of the wireless medium. In Section 1.1 of the previous chapter, we have described the three different types of wireless networks, namely infrastructured, infrastructureless and hybrid networks. In this section, we shall discuss the various types of wireless networks in more details.

2.4.1 Infrastructured Wireless Networks

IEEE 802.11 Wireless LAN (WLAN)

The IEEE 802.11 standard [IEEc], otherwise known as the Wi-Fi standard, denotes a set of standards for Wireless Local Area Network (WLAN). The original IEEE 802.11 standard [IEEc] in 1997 defines a common media access control (MAC) layer that supports the operation of all 802.11-based WLANs by performing core functions such as managing communications between radio network cards and access points. Subsequent amendments to 802.11 define specific physical (PHY) layers, such as 802.11b, 802.11g or 802.11a. The physical layer defines the data transmission for the WLAN, using various modulation schemes.

However, security in 802.11 WLAN is more complicated than in a wired environment. Furthermore, the two basic security mechanisms for secure access in 802.11 WLAN, namely entity authentication, including open system and shared key authentication and Wired Equivalent Privacy (WEP) have been shown to be vulnerable [CJL05]. To enhance security of IEEE 802.11, a new standard IEEE 802.11i [IEEa] has been developed. Besides introducing key management and establishment, 802.11i also defines encryption and authentication improvements. As WEP is vulnerable, 802.11i specifies enhanced encryption to provide stronger privacy. 802.11i also incorporates IEEE 802.1X [IEEb] as its authentication enhancement. 802.1X is widely deployed in many IEEE 802 series standards with the Remote Authentication Dial In User Service (RADIUS) [RWRS00] as the authentication server. But it was mentioned in [CJL05] that 802.1X is still unlikely to resolve all security threats in wireless networks. The 802.11i also defines a group key handshake that enables the authenticator (Access Point) to deliver the group transient key to the supplicant (user node)

so that the supplicant could receive broadcast messages. However, the group key handshake is a key distribution protocol and not a key agreement protocol. We refer readers to [CJL05] for more detailed discussions on Wireless LAN security issues.

Cellular networks

In cellular networks, a given area is divided into cells and each cell is under the control of a base station. Each base station handles a certain number of mobile clients that are in its immediate vicinity (i.e., communication between the mobile clients and the base station is single hop) and the base station plays an important role in the operation of the cellular network. Unlike Wireless Mesh Networks and ad hoc networks, cellular networks generally use licensed frequency bands (e.g. Global System for Mobile Communications (GSM)). For more information about cellular networks, we refer readers to [HRM02].

2.4.2 Infrastructureless Wireless Networks

Wireless Ad Hoc Networks

Unlike cellular networks that relies heavily on infrastructure (e.g. base stations), wireless ad hoc network is a network formed without any central administration [FJL00]. Wireless ad hoc network consist of nodes that can be mobile or stationary and these nodes use a wireless interface to send packet data. Since the nodes in wireless ad hoc networks can serve as routers and hosts, they can forward packets on behalf of other nodes and run user applications. Wireless ad hoc networks uses fully decentralized radio, access and routing technologies such as Bluetooth, packet radio network (PRnet) and IEEE 802.11 ad hoc mode to create different types of ad hoc networks for different applications. Examples of ad hoc networks include Mobile Ad hoc networks (MANETs), wireless sensor networks (WSNs) and personal area networks (PANs).

We discuss some of the characteristics of wireless ad hoc networks below taken from [FJL00]:

- **Distributed operation:** A node in an ad hoc network cannot rely on a network in the background to support security and routing functions. Instead, these functions must be designed so that they can operate efficiently under distributed conditions.

- **Dynamic network topology:** In general, if some or all the nodes in the ad hoc network are mobile, then the network topology will vary as the mobile nodes may move out of connectivity or leave the network at any time. These mobile nodes may also join the network at anytime they desire, which also changes the network topology. Therefore, connectivity in the network should be maintained to allow applications and services to operate undisrupted. In particular, this characteristic will influence the design of routing protocols.
- **Fluctuating link capacity:** the effects of high bit-error rates might be more profound in multihop ad hoc networks, since the aggregate of all link errors is what affects a multihop path. In addition, more than one end-to-end path can use a given link, which if the link is broken, then several sessions could be disrupted during the periods of high bit-error transmission rates. In this case, the routing function is also affected, however, efficient functions for link layer protection (such as forward error correction (FEC) and automatic repeat request (ARQ)) can substantially improve the link quality.
- **Low-power devices:** The nodes in wireless ad hoc networks are usually battery-operated and are often energy-constrained as a result. Therefore, the CPU computations for network applications should kept at a minimum while the transceiver input/output power for network applications should be optimized to consume as little energy as possible but still provide good communication performance. In addition, the introduction of multiple radio hops might improve overall performance besides achieving reasonable network connectivity. However, this can usually be realized only at the price of more complex routing.

We refer readers to [FJL00] for more detailed discussions on the wireless ad hoc networking.

Wireless Sensor Networks (WSNs)

Wireless Sensor Networks (WSNs), also known as distributed sensor networks (DSNs), are often used in military operations because WSNs can produce high-quality battlefield information using large numbers of physical sensors (e.g., acoustic, seismic, visual) communicating via ad hoc wireless networking. Advances in microelectromechanical systems (MEMS) technology allow sensors to be re-programmable in the battlefield, self-localizing, and to support low-energy, wireless, multi-hop

networking, while requiring only minimal pre-configuration [CKM00].

One primary application of WSNs in military operations is to detect and report events occurring within range of the WSN. Once an event such as movement of dismounted troops, movement of armored vehicles, detection of chemical occurrences, etc is detected, the detecting sensor node may report the event directly to a remote command and control application or collaborate with other sensors in the network to more reliably identify and track a target before reporting. Therefore, WSN communications (communications between sensor nodes, reporting of events to the remote command and control application or when accepting legitimate queries, commands and even software updates) have to be protected by encryption using a secret key (to provide confidentiality) because messages/data transmissions in WSNs are sent through radio waves that enable an adversary to eavesdrop on the communication, inject false messages and replay or change old messages. Furthermore, sensor nodes are usually deployed in hostile environments where they can be easily captured, compromised or manipulated by an adversary. Hence, key management and authentication are two essential security services to maintain proper operations of a WSN. Security protocols for WSNs based on pre-deployed keying (i.e. the key is issued before deployment) are easy to implement and have little overheads. However, most of these protocols based on pre-deployed keying have disadvantages regarding scalability and vulnerability to node capture. As compared to pre-deployed keying, a better key establishment technique for WSNs is the use of group key transport or group key agreement protocols where the nodes set up a shared secret key after deployment [GST07]. Group key agreement protocols are better solutions than group key transport protocols because in group key agreement protocols, no entity can predetermine the resulting key as it depends on the input of all participants.

Sensor nodes are the basic components of a WSN. Once the sensor nodes are deployed in battlefields, they usually have no mobility. Sensor nodes such as the Rockwell Scientific WINS nodes [ACPR99] are usually small, lightweight, inexpensive, low-power and battery-operated. Therefore, protocols designed for WSNs should consume as little energy as possible to extend the battery lifespan. Moreover, sensor nodes are usually placed in hostile environments and it is usually not possible to replace the batteries of sensor nodes when the batteries are depleted.

For more information about WSNs and sensor nodes, we refer readers to Chapter 2 of [CKM00].

2.4.3 Hybrid Networks

Wireless Mesh Networks

Like wireless ad hoc networks mentioned earlier, the nodes in wireless mesh networks (WMNs) act as a router or host for other nodes in the network. However, unlike wireless ad hoc networks, the nodes in WMNs can be fixed pieces of the network (infrastructure such as access points (AP) or base station (BS)) or mobile or stationary user nodes (infrastructureless such as laptop computers, PDAs and mobile phones). Hence, WMNs are hybrid networks where there can be devices from both infrastructured and infrastructureless networks.

It should be noted that wireless mesh networks is a type of network architecture and is not about which particular radio technology is used [Mot05]. Radio modulation determines how a specific radio will transmit and receive information over the wireless medium whereas network architecture defines the overall structure, components and interrelationships of devices in the network. Hence, WMNs can be applied to practically any radio schemes, effectively allowing the best radio to fit the desired application.

There are two types of nodes in WMNs [AWW05], namely the mesh routers such as access points (AP) or base station (BS) and the mesh clients such as laptop computers, PDAs and mobile phones. We describe these two types of nodes in WMNs below:

- **Mesh Routers:** The mesh routers are fixed pieces of the network and do not have any energy constraints as they are usually connected to a fixed power supply. Besides routing capabilities for gateway/repeater functions as in a conventional wireless router, a wireless mesh router contains additional routing functions to support mesh networking. To further improve flexibility of mesh networking, mesh routers is usually equipped with multiple wireless interfaces built on either the same or different wireless access technology. Compared to a conventional wireless router, a wireless mesh router can achieve the same coverage with much lower transmission power through multi-hop communications. The medium access control (MAC) protocol in a mesh router can also be enhanced with better scalability in a multi-hop mesh environment. Despite all these differences, mesh routers and conventional wireless routers are usually built based on a similar hardware platform. Mesh routers can be built based on

dedicated computer systems (e.g. embedded systems) and look compact or they can be built based on general-purpose computer systems such as laptop or desktop computers [AWW05].

- Mesh clients: Unlike mesh routers, mesh clients are usually battery-operated devices with energy-constraints. Mesh clients usually have only one wireless interface. As a result, the hardware platform and software for mesh clients can be simpler than those for mesh routers. As compared to mesh routers, mesh clients have a higher variety of devices. They can be laptop/desktop computers, pocket PC, PDAs, IP phone, RFID reader, and many other devices [AWW05]. However, similar to the nodes in wireless ad hoc networks, mesh clients may require energy-efficient protocols in order to prolong its battery lifespan. Hence CPU computations for network applications in mesh clients should kept as minimum as possible and the communications protocols should be optimized to consume as little energy as possible but still provide good communication performance.

In a typical infrastructured wireless networks such as WLAN, Radio Frequency (RF) signal attenuation can impose unacceptable tradeoffs between power distance and throughput, disrupt data streams as users cross roaming boundaries and even lead to dropped connections when physical barriers (e.g. objects that absorb or reflect signal strength) are present. All of these limitations result from the way signal strength drops exponentially as a function of distance between sender and receiver. WMNs solve this problem with the use of multi-hopping [KCY⁺02], which achieves higher throughput without sacrificing effective radio range via shorter link distances, less interference between the nodes and more efficient frequency re-use. Multi-hopping makes routing in WMNs a very important and necessary functionality. Therefore, the routing mechanism in WMNs must be secure as adversaries may be tempted to attack it. We refer readers to [AWW05, SH06, Mot05] for more detail discussions of WMNs.

2.4.4 Inherent Characteristics of Wireless Networks

In this section, we discuss some of the specific characteristics of wireless network environment that play an important influence in the design of efficient, scalable and authenticated GKM protocols for wireless networks. These inherent characteristics that pose challenges to GKM protocols the various types of wireless networks include:

Ease of eavesdropping As communications in wireless networks is sent on the wireless channel, it is fairly easy for anyone with the right kind of radio to eavesdrop on the wireless communication unless in a wired network where the eavesdropper needs to do some work (e.g. wire-tapping) to be able to eavesdrop in the communication.

Lack of computational powers Most user devices in wireless networks are typically low power energy constrained devices such as PDAs which have much less computational capabilities than an average desktop system.

Limited battery (energy) resources Wireless devices such as PDAs and laptop computers are usually powered by batteries which is a very limited resource. In most cases, once the batteries on the wireless device runs out, the user would have to manually replace or recharge the batteries and communications may be interrupted especially when replacing batteries. Furthermore, in certain wireless sensor networks, it may not even be possible to replace batteries. Therefore, there is a need to design energy-efficient GKM protocols for wireless networks to prolong the battery lifespan of wireless devices.

Communication constraints Communication in wireless networks consume quite a large amount of battery power and furthermore, wireless communications have limited transmission range.

Lack of central infrastructures In certain types of wireless networks such as ad hoc networks, there is very little or no central infrastructure to make use of.

Scalability In some large scale wireless networks such as wireless mesh networks or wireless sensor networks, there might be hundreds or thousands of nodes. Therefore, GKM protocols for wireless networks must be scalable to ensure nodes in such large-scale wireless networks do not deplete a large amount of energy when running GKM protocols.

Highly dynamic topology Wireless networks have network topologies that are highly dynamic as user nodes which can be mobile can join and leave the wireless network at any time.

Denial-of-Service GKM protocols for wireless networks are vulnerable to DoS attacks. This is especially true in wireless networks such as WMNs where there is a certain type of infrastructure, i.e. there are high power nodes such as mesh routers that can act as controllers in the GKM protocol to ease the computational burden of lower power user nodes. In such GKM protocols with controller nodes (e.g. NLKW [NLKW05], NLKKW [NLK⁺04] and NHS [NKW05]), the adversary may launch DoS attacks by sending invalid Start GKA or Join request and force the controller nodes to constantly perform computations, thus depleting the controller nodes' computational resources and preventing legitimate group members from sending legitimate Start GKA or Join requests to the controller nodes. Hence there is a need for GKM protocols to provide protection against DoS attacks.

The challenges mentioned above, some unseen in existing collaborative group applications for wired networks (e.g. Internet) have resulted in some current GKM protocols to be unsuitable or inefficient for use in wireless networks. Therefore, there is a need to design energy-efficient and suitable GKM protocols that address the above-mentioned challenges in wireless networks.

2.4.5 Heterogeneous and homogeneous Wireless Networks

Wireless networks can also be classified as either heterogeneous or homogeneous wireless networks depending on the type of nodes in the network. We explain these type types of wireless networks as follows:

Heterogeneous Wireless Networks refer to wireless networks that consist of nodes with different performance capabilities (i.e. some nodes are more powerful than other nodes). Hybrid networks are examples of heterogeneous wireless networks. For example, in Wireless Mesh Networks (WMNs), which is a type of hybrid network, the mesh routers are high power nodes without energy constraints while the mesh clients are low power energy constrained devices. A *static* heterogeneous wireless network consists of stationary network nodes that do not move while a *dynamic* heterogeneous wireless network consists of both mobile nodes (e.g. PDAs) and stationary nodes that do not move (e.g. mesh routers).

Homogeneous Wireless Networks refer to wireless networks where all nodes in the network have the same performance capabilities. Infrastructureless wireless networks such as ad hoc networks are examples of homogeneous wireless networks. For example, in certain types of ad hoc networks, all network nodes are low-power energy constrained devices (e.g. PDAs). Another example is wireless sensor networks where all nodes are low-power energy constrained devices (e.g. Rockwell Scientific Wireless Integrated Network Sensor (WINS) nodes). A *static* homogeneous wireless network consists of stationary network nodes that do not move (e.g. in certain types of wireless sensor networks) while a *dynamic* homogeneous wireless network consists of mobile nodes (e.g. PDAs and laptop computers).

2.5 Key Establishment Concept

In most cryptographic operations, it can be seen that the key is an essential component. Therefore, Key Management, which is the set of techniques and procedures to support the establishment and maintenance of keys between authorized parties, is a very important task. In this section, we shall give an overview of key establishment and also discuss on the different types of attacks from different adversaries. Key Establishment is defined as a process or protocol whereby a shared secret key becomes available to two or more parties for subsequent cryptographic use. Key establishment can be divided into two categories : Key transport and key agreement.

Key Transport protocols or mechanisms are key establishment techniques where one party creates or otherwise obtains a secret value and securely transfers this secret value to the other parties.

Key Agreement protocols or mechanisms are key establishment techniques where the shared secret is derived by two or more parties as a function of the information contributed by, or associated with, each of these such that no one party can predetermine the resulting secret value.

Key agreement protocols have the advantage over key transport protocols because each user does not have to rely on any other party to generate the keys. Therefore, if the nodes in the key agreement protocol are not malicious, it can often be guaranteed that the session key is sufficiently

random if at least one of the parties is able to generate sufficiently random inputs [BM03]. A relative benefit is that users can usually ensure that the session key is fresh by ensuring that their own input is fresh. Furthermore, in key transport protocols, the party that is responsible for creating the secret value acts as a single point of failure for the key transport protocol.

2.5.1 Type of Adversary

When examining the security of key establishment protocols, it is assumed that the adversary attempts to attack the protocol itself and not the underlying cryptographic mechanism used such as encryption algorithm and digital signature schemes which are assumed to be secure.

An **adversary** is defined as an entity which is neither the sender nor receiver in the communication and which tries to defeat the information security service being provided between the sender and receiver. An adversary will often attempt to play the role of either the legitimate sender or legitimate receiver.

The adversary may launch either a passive attack or active attack. **Passive attack** involves an adversary who eavesdrop on the communication and analyzes the captured messages while an **active attack** involves an adversary who modifies/injects messages into the communication.

The adversary or attacker can be further distinguished as outsider or insider adversary.

An **outsider** is an adversary with no special knowledge beyond that generally available while an **insider** is an adversary with access to additional information (e.g. session keys or secret partial information) obtained by some privileged means (e.g. conspiracy, access to legitimate user devices).

2.5.2 Diffie-Hellman Key Agreement

We now describe a basic Diffie-Hellman (DH) key agreement protocol for two parties proposed by Diffie and Hellman in [DH76]. Two parties Alice (A) and Bob (B) first agree on two appropriate primes p and q such that q divides $p - 1$. (p is normally 1024 bits and q is normally 160 bits) and a generator g of Z_p^* . They perform the following actions:

1. A selects random $x \in Z_q$, computes and sends $g^x \bmod p$ to B .
2. B selects random $y \in Z_q$, computes and sends $g^y \bmod p$ to A .

3. A then computes the shared key $K = (g^y)^x = g^{xy} \bmod p$ and B computes the shared key $K = (g^x)^y = g^{xy} \bmod p$.

Hence, both A and B have a common secret key K that can be used to secure subsequent communications between them. The security of K lies with the hardness of the Discrete Logarithm Problem. However, without authentication, the two party DH key agreement protocol described above is susceptible to a man-in-the-middle attack where an adversary Carol C can intercept messages from both A and B and change them to establish individual keys with both A and B , while both A and B think that they have established a key with each other. To prevent such an active attack, both A and B need to authenticate each other by means of digital signatures.

2.5.3 Authentication and Key Exchange Protocols

Authentication and key exchange protocols for wireless networks have been important research topics in recent years. The IEEE 802.11 [IEEc] specified the Wired Equivalent Privacy (WEP) protocol for authentication and key exchange in wireless local area networks (WLAN). However, it was pointed out in [ASWZ02, BGW01] that WEP is insecure. To solve the problems in WEP, IEEE specified the IEEE 802.1X standard [IEEb], which uses the Extensible Authentication Protocol (EAP) to enhance authentication and key exchange for WLAN. However, authentication methods defined for EAP such as EAP-MD5, EAP-TLS (Transport Layer Security), EAP-TTLS (Tunnelled TLS) and PEAP (Protected EAP) were pointed out in [WDBA05] to have limitations and cannot fulfill the security requirements of wireless networks. Funabika et al [FNT⁺06] proposed an anonymous IEEE 802.1X authentication protocol, but their protocol requires the clients to hold membership certificates and does not provide security against DoS attacks.

Password-based public-key cryptographic protocols such as PAK [Mac02], SRP [Wu98] and SPEKE [Jab96] have also been used to create EAP authentication methods such as EAP-PAK, EAP-SRP and EAP-SPEKE respectively by the IEEE P1363 Standard Working Group. However, these password-based EAP authentication methods do not provide client anonymity and are susceptible to DoS attacks. Wan et al [WDBA05] pointed out the security flaws in the Lancaster protocol [SFW⁺01] and proposed a DoS-Resistant anonymous access control protocol, which is also considered as the authentication and key exchange protocol for accessing wireless networks

(For brevity, we shall denote access control protocol as authentication and key exchange protocol throughout this thesis). Liu et al [LTL⁺05] also proposed another password-based authentication protocol for access control for WLAN, however their protocol does not provide client anonymity nor protection against DoS attacks. Wang [Wan04] proposed an anonymous authentication protocol that concentrated mainly on cellular networks but the protocol does not provide protection against DoS attacks. Wong [Won05] showed that two anonymous authentication protocols, namely Varadharajan-Mu protocol [VM97] and Go-Kim protocol [GK01], are vulnerable to practical attacks and the anonymity of the client could be compromised.

Review of Wan et al's Authentication and Key Exchange Protocol

We shall now review Wan et al's Authentication and Key Exchange Protocol [WDBA05] that consists of five rounds and comprises of client C, access point AP and an authentication server S. The client C communicates with S through AP. The protocol works as follows:

Round 1: C first selects random $x \in Z_q^*$, computes $z_c = g^x \bmod p$ and encrypts z_c using password pw and obtains $z_c^* = \mathcal{E}_{pw}(z_c)$. C then sends z_c^* to S.

Round 2: Upon receiving the first message above, S chooses random $y \in Z_q^*$, computes $z_s = g^y \bmod p$ and a hash value $\nu = H_{HK}(z_s, z_c^*)$ with hash key HK private to S only. Finally, S sends $\{S, z_s, z_c^*, \nu\}$ to C.

Round 3: After C receives the above message, C computes session key $k = H(C, S, z_s^x)$ and $\theta_c = H_k(z_s, z_c^*)$. Next, C selects a random nonce N_c and encrypts $\{C, MAC_C, N_c, \theta_c\}$ using S's public key $p\kappa$ and obtains $c_s = E_{p\kappa}(C, MAC_C, N_c, \theta_c)$. Finally, C sends $\{z_s, z_c^*, \nu, c_s\}$ to S.

Round 4: After S receives the above message, S verifies the validity of ν . If ν is valid, S decrypts c_s and fetches C's password pw according to his identity. S then decrypts z_c^* and obtains z_c and computes session key $k = H(C, S, z_c^y)$. Next, S checks θ_c . If θ_c is valid, S assigns an IP address IP_C to C and encrypts $\{C, S, IP_C, z_s, z_c^*\}$ using session key k and obtains $c'_s = \mathcal{E}_k(C, S, IP_C, z_s, z_c^*)$. Finally, S sends c'_s to C.

Round 5: S sends session key k , client's MAC address MAC_C and IP address IP_C to AP through a secure channel.

2.6 Group Key Agreement Protocols

As wireless communications and information services become more and more sophisticated, many applications involving multiple users have become necessary. As these applications will generally require secure communications, it is necessary to design protocols that establish keys for groups of user nodes. Depending on the application, different practical requirements are considered when designing group key management protocols. The main objective of this thesis is to examine different requirements in wireless networks for secure group communications and design secure and efficient group key management protocols. Once a common key is shared by group members, confidentiality can be provided in the secure group communication by encrypting every message using the symmetric key encryption scheme with the established group key. In this section, we will examine the ways in which the two-party key management protocols can be extended to the multi-party or group key management protocols.

We first provide the definition of group key management and group key management as follows:

Definition 4 *Group Key Establishment* is a process whereby a shared secret key is established and becomes available to a group of users for subsequent cryptographic use.

Definition 5 *Group Key Management* is the set of processes and mechanisms that support group key establishment and the maintenance of ongoing keying relationships between group parties including replacing older keys with new keys when necessary.

In this thesis, we are interested in **Group Key Management** and **Group Key Agreement** in particular.

Definition 6 *A group key agreement protocol is a group key management protocol where the group key K is generated using the random inputs (key shares) of all the group members. The group key agreement protocol must ensure that*

- *No information about K can be extracted from the protocol run without the knowledge of at least one of the random inputs (key shares).*

- All random inputs (key shares) by group members are kept secret, that is, a collusion of group members cannot obtain any information of the random input (key share) of an honest group member.

On the other hand, there is another group key management technique known as Group Key Distribution. The definition of group key distribution protocol is as follows:

Definition 7 *A group key distribution protocol is a group key management protocol whereby one dedicated party (referred as a controller) chooses the group key and distributes it to all group members using a secure channel.*

Group Key Distribution can be classified as *decentralized* group key distribution if the key server is dynamically selected or *centralized* group key distribution if the key server is predetermined. Group key distributions have the disadvantage that the key server is a single point of failure and nodes in the group have to rely on the key server to generate good (cryptographically strong and random) keys.

2.6.1 Security Goals for Group Key Management

One of the most important security requirements of group key management is called **key freshness**. A key is fresh if it can be guaranteed to be new as opposed to possibly an old key being reused by an adversary. Furthermore, a session key should only be known to the involved parties (**key authentication**). A secure group key management protocol should also provide the following security properties:

1. **Group Key Secrecy:** This property guarantees that it is computationally infeasible for an active adversary to discover the derived group key.
2. **Forward Secrecy:** Forward secrecy is to prevent a leaving or expelled group member from obtaining the new group key formed by the remaining members for future secure group communications.
3. **Backward Secrecy:** Backward secrecy is used to prevent new group members from obtaining the previous group keys even though the new group member has established a new group key with existing group members.

4. **Key Independence:** An adversary who knows a subset of group keys in the life-cycle of a group should not be able to discover any other group keys outside this subset.

Next, we introduce the definitions of key authentication and key confirmation related to authenticated group key management from [MOV97] and [BM03].

Definition 8 *Key authentication* is the property that the established group key is known only to the honest group members taking part in the group key agreement.

Many group key agreement protocols simply allow each group member to implicitly authenticate the group key with respect to only one other group member [BM03]. For example, in group key agreement protocols, group members usually communicate with each other in some systematic way. Hence group key agreement protocols with members that are arranged in a logical ring can authenticate each member's key input to the 'next' member. Alternatively, if there is a controller in the group key agreement protocol, then the group members can authenticate the key with the controller. Key confirmation is the property that all group members are assured that other group members have the same key as himself. The definition of key confirmation in [BM03] is as follows:

Definition 9 *Key confirmation* Let $\mathcal{U}$ be a set of group members with $U_i, U_j \in \mathcal{U}$. Key confirmation of U_j to U_i is provided if U_i has assurance that group key K is a valid group key to communicate with every group member in $\mathcal{U}$ and U_j has possession of K . Complete key confirmation is provided to U_i if key confirmation of U_j is provided to U_i for all $U_j \in \mathcal{U} \setminus \{U_i\}$.

2.6.2 Static and Dynamic Group

After a group key is established for a group of users, it may be desirable to add or remove users from the group depending on the underlying group communication application. On the other hand, in some group settings, the users are fixed and no adding or removal of group members are necessary. In this section, we shall examine the different types of group setting possible for group key agreement protocols in wireless networks.

Types of Group Groups can be classified into two types, namely **static** and **dynamic** groups. A *static* group is one in which there are no group membership events such as user join, user leave

or subgroup merge. *Dynamic* groups are groups where nodes or users arbitrarily join or leave the network at any time. Hence, the network architecture for these groups usually change frequently. In order to provide secure group communication for dynamic groups, the group key agreement protocol used has to consider group membership events and update the group key accordingly.

2.7 Examples of Group Key Management Protocol

In this section, we give a literature review on several current group key agreement protocols that have been proposed in this area of research. We shall also present GKA protocols that uses different types of authentication such as identity-based GKA protocols and password-based GKA protocols. Unless otherwise stated, two random large primes, q and p , are selected such that q divides $p - 1$ (p is normally 1024 bits and q is normally 160 bits) and an element $g \in Z_p^*$ of order q is chosen.

2.7.1 ING Protocol

In the ING [ITW82] protocol, all members U_i for $1 \leq i \leq n$ must be arranged in a logical ring. In the first round, every member U_i generates its session random r_i and computes g^{r_i} . At each round, each member raise the value received in the previous round to its session random r_i and sends it to the next node in the sequence. Therefore, n messages are exchanged among the group members at each round. After $n - 1$ rounds, every member computes the same group key $K_n = g^{r_1 r_2 \dots r_n}$. The main disadvantages of this protocol are that it requires a high number of messages to be exchanged, $n - 1$ rounds to compute the group key and n sequential modular exponentiations are required.

2.7.2 Hypercube and Octopus Protocols

The Hypercube protocol was presented by Becker and Willie [BW98] with the intentions to overcome the high number of messages needed by the ING protocol by arranging the nodes in a hypercube. For a network shown in Figure 2.2 consisting of four nodes positioned as a square, a key is established between A and B ($g^{r_A r_B}$) and another key between C and D ($g^{r_C r_D}$) first, then these keys are used to establish the final group key $K = g^{r_A r_B g^{r_C r_D}}$ amongst the four users.

The Octopus protocol is an extension of the Hypercube protocol for networks with an arbitrary

number of nodes. The central control comprises of a hypercube and the rest of the group members are divided into subgroups with each subgroup having a subgroup controller that is part of the central control. Figure 2.3 displays an Octopus protocol with group size $n = 13$, with nodes A, B, C and D belonging to the central control (Hypercube). First, the members that do not belong to the central control will perform a DH key exchange with their respective subgroup controller. Next, the subgroup controllers establish a group key in the central control using the hypercube protocol. The group key is then distributed to the other nodes.

Figure 2.2: Hypercube protocol for 4 users

Figure 2.3: Octopus protocol for 13 users

2.7.3 Group Diffie-Hellman Protocol

The Group Diffie-Hellman (GDH) protocols, also known as CLIQUES protocols, are a suite of group key management protocols developed by Steiner et al [STW96,STW98] for dynamic groups. Two of these protocols Initial Key Agreement.1 (IKA.1) and IKA.2 are defined for group generation while the other protocols are specified for dynamic group membership events such as join, leave and subgroup merge. IKA.2, an improvement of IKA.1, minimizes the number of computation performed by each member in IKA.1.

IKA.2 The IKA of GDH consist of two stages, namely upflow and downflow. In the upflow stage, the first member U_1 starts the group key agreement by selecting a random secret r_1 , computes g^{r_1} and unicast it to the next member, U_2 . U_2 will selects his random secret r_2 , raise the received message to r_2 getting $g^{r_1 r_2}$ and send this value to the next member, U_3 . This process repeats for each member U_i until it reaches U_{n-1} . U_{n-1} will selects his random secret r_{n-1} , raise the received

message to r_{n-1} getting $g^{r_1 r_2 \dots r_{n-1}}$ and broadcast this value to all. Upon receiving the message from U_{n-1} , members U_i for $1 \leq i \leq n-2$ would factor out their contribution (by raising the received message to the inverse of their secret) and send it to U_n . U_n (also known as the controller) would then perform the downflow stage by taking the values received from each U_i for $1 \leq i \leq n-1$, raise it to his own random r_n and then broadcast the result to the group. Each member U_i picks up the value meant for him (i.e. $g^{r_1 r_2 \dots r_n / r_i}$) and raise it to his random secret r_i to obtain the final group key $K = g^{r_1 r_2 \dots r_n}$.

Join/Merge When a new member U_{n+1} arrives, the current controller U_n generates a new random r'_n and forwards an “IKA.2-upflow like” message to U_{n+1} . U_{n+1} becomes the new controller and will select a random, r_{n+1} and broadcast an “IKA.2-downflow like” message to the group. All group members will then use the technique mentioned in IKA.2 to compute the final group key K .

A Merge can be considered as the joining of k new members. Like the Join event, U_{n+1} first receives a “Join-response like” message from the old controller U_n and thereafter executes the IKA.2 protocol amongst the new members. Finally, U_{n+k} becomes the new controller and broadcast a downflow message which will enable all members (old and new) to compute the new group key.

Leave In a leave event, the remaining highest-indexed member will be appointed as the controller. This new controller will choose a new random secret, raise all the stored information that were sent by non-leaving members to this new random and broadcast a downflow message with values that only allows the non-leaving members to compute the new group key. For example, in a group of 5 members, suppose the fourth member U_4 leaves, since U_5 is the highest indexed remaining member, U_5 becomes the controller and selects a new random r'_5 . Next, U_5 raises all the stored information that were received from non-leaving members $\{U_1, U_2, U_3\}$ to r'_5 and broadcast the new information to the remaining members $\{U_1, U_2, U_3\}$. It should be noted that U_5 does not broadcast rekeying message to leaving member U_4 . This will ensure that the leaving member U_4 is unable to compute the new group key.

2.7.4 Tree-Based Group Diffie-Hellman Protocol

The Tree-based Group Diffie-Hellman (TGDH) [KPT00,KPT04b] combines the concepts of binary key tree structure similar to those proposed in [WHA98,WGL00] with the DH key exchange. We first explain the Initial key agreement (IKA) with an example below.

Figure 2.4: Key Tree in TGDH

Figure 2.4 above shows an example of a key tree with 4 members, U_1, U_2, U_3 and U_4 . Every node in the binary key tree is either a leaf node or a parent of two nodes. Each leaf node will host a member U_i , for example in the figure above, node $(2, 2)$ is hosting member U_3 . Each node is denoted (l, v) where l denotes the level and $0 \leq v \leq 2^l - 1$ denotes the number of nodes in a level. Each node, (l, v) is associated with a random secret $K_{(l,v)}$ and a blinded key, $BK_{(l,v)} = f(K_{(l,v)})$ where the function $f(k) = g^k \mod p$. While $K_{(l,v)}$ is secret, $BK_{(l,v)}$ is public. A member U_i at node (l, v) can compute every key along its path from (l, v) to $(0, 0)$, referred to as the *key-path* (KEY_i^*). Every key $K_{(l,v)}$ is computed using the following formula:

$$\begin{aligned}
 K_{(l,v)} &= (BK_{(l+1,2v+1)})^{K_{(l+1,2v)}} \mod p \\
 &= (BK_{(l+1,2v)})^{K_{(l+1,2v+1)}} \mod p \\
 &= g^{K_{(l+1,2v)} K_{(l+1,2v+1)}} \mod p
 \end{aligned}$$

In other words, computing a key at (l, v) requires the knowledge of the secret key of one of the two children nodes and the blinded key of the other child node. $K_{(0,0)}$ is the group key that is shared by all members.

For example, in Figure 2.4 above, to compute the final group key $K_{(0,0)}$, U_3 first selects a random secret r_3 and set $K_{(2,2)} = r_3$. Next, U_3 computes $K_{(1,1)}$ by raising $BK_{(2,3)} = g^{r_4}$ to his random secret $K_{(2,2)}$, i.e. $K_{(1,1)} = (BK_{(2,3)})^{K_{(2,2)}} = g^{r_3 r_4}$. U_3 then computes $K_{(0,0)} = (BK_{(1,0)})^{K_{(1,1)}} = g^{g^{r_1 r_2} g^{r_3 r_4}}$ since $BK_{(1,0)} = g^{g^{r_1 r_2}}$. The term co-path (CO_i^*) is defined as the set of siblings of each node on the key-path of member U_i . For example, the co-path (CO_3^*) of member U_3 in the figure above is the set of nodes $(2,3), (1,0)$. It can be seen that every member U_i at leaf node (l, v) can derive the group key $K_{(0,0)}$ from all blinded keys on the co-path (CO_i^*) and its secret key $K_{(l,v)}$.

Join/Merge A new member U_{n+1} initiates the Join protocol by sending a Join request message containing its blinded key $BK_{n+1} = g^{r_{n+1}}$. When the current group members receive this message, they determine the insertion node in the tree. The insertion node is the shallowest rightmost node, where the Join does not increase the height of the key tree. Otherwise, U_{n+1} will join to the root node if the key tree is well balanced. A sponsor will be appointed, and the sponsor is the rightmost leaf node in the subtree rooted at the insertion node. Next, the sponsor creates a new intermediate node and a new member node, and promotes the new intermediate node to be the parent of both the insertion node and the new member node. After updating the tree, the sponsor computes the new group key (since it knows all the necessary blinded keys) and broadcasts the new tree containing all blinded keys. All other members update their tree accordingly and compute new group key. A merge is similar, where instead of one member, we have a smaller tree joining the larger one.

Leave Suppose a member U_d wishes to leave the group. In this case, a sponsor, which is the rightmost leaf node of the subtree rooted at the leaving member's sibling node, is appointed. In the Leave protocol, every member updates its key tree by deleting the leaf node corresponding to U_d . The former sibling of U_d is promoted to replace U_d 's parent node. The sponsor picks a new random secret, computes all the keys on its key path up to the root and broadcast the new set of blinded keys to the group. This information allows all remaining members to recompute the new group key K' .

2.7.5 Skinny Tree Protocol

The Skinny Tree (STR) protocol, also designed by Kim et al. [KPT01, KPT04a], is similar to the TGDH protocol mentioned above but differs only in the nature of the key tree being used. While TGDH uses a binary key tree, STR employs a special type of tree in which each node is either a leaf node or has two child nodes attached. Figure 2.5 shows an example of a STR key tree.

Figure 2.5: A five member STR Key Tree

In Figure 2.5, we have a STR key tree hosting five members at each leaf position. The tree has two types of nodes: leaf and internal. Each leaf node is associated with a specific group member. An internal node IN_i for $1 \leq i \leq n$ where n is the group size always has two children: another (lower) internal node IN_{i-1} and a leaf node LN_i . The exception is IN_1 , which is also a leaf node LN_1 corresponding to U_1 . Computing the final root (group) key K is similar to that in TGDH. Thus, to compute the group key, every member node U_i has to know the blinded key (BK) of its peer intermediate node and the higher member nodes. The group key for this tree is associated with the root node, IN_5 , and is given as $K_{IN_5} = g^{r_5 g^{r_4 g^{r_3 g^{r_1 r_2}}}}$.

Join/Merge The new joining member U_{n+1} simply joins at the root of the existing tree. The highest member in the existing tree acts as a sponsor and creates a new root node and makes the incoming node as the new root node's right child and the current existing tree its left child. It also changes its random secret and broadcast the new blinded keys to the group. Merge is similar to Join. The smaller of the two trees joins the larger tree on the top. The highest-indexed member of the larger tree acts as a sponsor and broadcasts the new blinded keys.

Leave When a Leave event occurs, the leaving node and its intermediate parent node are deleted from the tree. The member just below the leaving member acts as the sponsor. It changes its random secret and broadcasts the new blinded keys to all remaining users. Upon receiving the new set of blinded keys, all remaining members are now able to compute the new group key K' .

2.7.6 Burmester-Desmedt Group Key Agreement Scheme

In 1994, Burmester and Desmedt proposed an efficient and secure Burmester-Desmedt (BD) group key agreement protocol [BD95] that requires only *two* rounds of communication and only *one* message to be broadcast by each member in each round, regardless of group size n . The BD protocol utilizes a so called “ring-based structure”, where members are arranged in a ring, i.e. the member after the last member U_n is again U_1 . Let p and q be two primes such that q divides $p - 1$ (p is normally 1024 bits and q is normally 160 bits). Next, an element $g \in Z_p^*$ of order q is chosen. The BD Protocol works as follows:

In Round 1, each member U_i for $1 \leq i \leq n$ selects random secret $r_i \in Z_q^*$ and broadcasts $z_i = g^{r_i} \bmod p$. In Round 2, all members compute and broadcast X_i , where

$$X_i = \left(\frac{z_{i+1}}{z_{i-1}} \right)^{r_i} = g^{r_i r_{i+1} - r_{i-1} r_i} \bmod p. \quad (2.2)$$

After two communications rounds, each U_i for $1 \leq i \leq n$ computes the common group key K as follows:

$$K = (Z_{i-1}^{r_i})^n \cdot X_i^{n-1} \cdot X_{i+1}^{n-2} \cdots X_{i-2} \bmod p = g^{r_1 r_2 + r_2 r_3 + \dots + r_n r_1} \bmod p \quad (2.3)$$

The paper [BD95] did not provide protocols to handle dynamic group membership events such as join, leave and merge. One way to handle such events as pointed out in [AKNR⁺04] and [KPT00, KPT04b] is to re-execute the BD protocol whenever a dynamic group membership event occurs. Steiner et al [STW98] briefly described another technique for adding and removing members for BD. Although their technique is more efficient than re-executing the BD protocol, we show in this thesis that their technique does not provide backward secrecy (a joining member is able to find out the earlier key used). The BD GKA protocol has been proven secure by Katz and Yung [KY03] under the Decisional Diffie-Hellman (DDH) assumption. Hence we state the following theorem:

Theorem 1 *The BD GKA protocol is secure under the DDH assumption.*

We refer readers to [KY03] for the proof of this theorem.

2.7.7 Nam et al DDH-based Group Key Agreement Protocol

Nam et al [NLKW05] proposed a constant-round GKA protocol (NLKW). In the NLKW protocol, there is a controller node U_1 that performs most of the computation while other users U_i for $2 \leq i \leq n$ only performs two modular exponentiations. The NLKW protocol does not provide dynamic protocols such as Join and Leave Protocols to handle join and leave events. Two random large primes, q and p , are selected such that q divides $p - 1$. (p is 1024 bits and q is 160 bits.) Next, an element $g \in Z_p^*$ of order q is selected. Let H be a one way cryptographic hash function where $H : \{0, 1\}^* \rightarrow \{0, 1\}^\lambda$, where λ is an integer. The NLKW protocol is described as follows:

In Round 1, all users U_i for $2 \leq i \leq n$ selects random $r_i \in Z_q^*$, computes $z_i = g^{r_i} \bmod p$ and sends $m_i = U_i || z_i$ to controller node U_1 .

In Round 2, U_1 first selects $r, r_1 \in Z_q^*$ and computes $z = g^r \bmod p$ and $y_1 = g^{r r_1} \bmod p$. U_1 then computes $y_i = z_i^r \bmod p$ for $2 \leq i \leq n$, $Y = \prod_{i=1}^n y_i \bmod p$, $T_i = Y \cdot y_i^{-1} \bmod p$ and the set $T = \{T_i | 2 \leq i \leq n\}$. Next, U_1 broadcasts $m_1 = U_1 || z || T$ to the group.

In the key computation round, upon receiving the broadcast message, each user U_i computes $Y = z^{r_i} \cdot T_i \bmod p$. All users U_i for $1 \leq i \leq n$ then compute the session key as $K = H(T || Y)$.

The NLKW protocol is proven to be secure under the decisional Diffie-Hellman (DDH) assumption by Nam et al [NLKW05]. Hence we list the following theorem:

Theorem 2 *The NLKW protocol is secure provided the DDH problem is hard.*

We refer readers to [NLKW05] for details of the security proof.

2.7.8 Nam et al Dynamic Group Key Exchange Protocol

Nam et al [NLK⁺04] proposed another communication-efficient GKA protocol (NLKKW) that is similar to the NLKW protocol [NLKW05] described in Section 2.7.7. In the NLKKW protocol, there is also a controller U_1 that performs most of the computations while other group members U_i for $2 \leq i \leq n$ only performs two modular exponentiations. An important difference between the

NLKKW protocol [NLK⁺04] described in this section and the NLKW protocol [NLKW05] described in Section 2.7.7 is that the NLKKW protocol [NLK⁺04] is based on the difficulty of the factoring problem while the NLKW protocol [NLKW05] is based on the difficulty of the DDH problem.

Nam et al [NLK⁺04] introduced three protocols for NLKKW: Initial GKA Protocol, Leave Protocol and Join Protocol for initial group formation, user leave and user join respectively. These three NLKKW protocols work on Z_N^* , the multiplicative group modulo $\bar{N}$, where $\bar{N}$ is the product of two large distinct primes $\bar{p}$ and $\bar{q}$ of equal length such that $\bar{p} = 2\bar{p}' + 1$ and $\bar{q} = 2\bar{q}' + 1$, where $\bar{p}'$ and $\bar{q}'$ are also prime integers. Then $\bar{N}$ is a Blum integer since $\bar{p} \equiv 3 \pmod{4}$ and $\bar{q} \equiv 3 \pmod{4}$. An element $v \in Z_N^*$ is called a quadratic residue modulo $\bar{N}$ if there exist an $x \in Z_N^*$ such that $x^2 \equiv v \pmod{\bar{N}}$. If no such x exists, then v is called a quadratic non-residue modulo $\bar{N}$. $g \neq 1$ is denoted as a quadratic residue that is chosen uniformly at random in the set of quadratic residues in Z_N^* . Using this quadratic residue g , a finite group $G = \langle g \rangle$ is defined where $\langle g \rangle$ is the cyclic subgroup of Z_N^* generated by g . Let H be a one way cryptographic hash function where $H : \{0, 1\}^* \rightarrow \{0, 1\}^\lambda$, where λ is an integer. In the NLKKW protocol, all messages sent are signed by the sender and all messages received are verified to ensure that the group key agreement is authenticated.

Initial GKA Protocol Assume we have a group of n users, $\mathcal{G} = \{U_1, \dots, U_n\}$ who wish to establish a group key by running the Initial GKA Protocol protocol where U_1 is the controller node. The Initial GKA Protocol protocol runs in two rounds and is described as follows:

During Round 1, all users U_i for $2 \leq i \leq n$ selects random $r_i = \{1, 2, 3, \dots, \bar{N}\}$ and computes $z_i = g^{r_i} \pmod{\bar{N}}$. Next U_i sends z_i with signature σ_i to controller U_1 . In Round 2, after receiving and verifying all received messages, U_1 first selects r_1 and compute $z_1 = g^{r_1} \pmod{\bar{N}}$. U_1 computes $y_i = z_i^{r_1} \pmod{\bar{N}}$. U_1 then computes Y as $Y = \prod_{i=2}^n y_i \pmod{\bar{N}}$ if n is even or as $Y = \prod_{i=1}^n y_i \pmod{\bar{N}}$ if n is odd. U_1 also computes the set $T = \{T_i | 2 \leq i \leq n\}$ where $T_i = Y \cdot y_i^{-1} \pmod{\bar{N}}$ for $2 \leq i \leq n$. Let $Z = \{z_i | 1 \leq i \leq n\}$. Next, U_1 broadcast Z , T and σ_1 to the group, where σ_1 is the signature of the broadcast message. During the key computation round, upon receiving and verifying the message from U_1 , each user U_i computes $Y = z_1^{r_i} \cdot T_i \pmod{\bar{N}}$. All users U_i for $1 \leq i \leq n$ then compute the session key as $K = H(T||Y)$ and stores their random secret r_i and all received z_i for future use.

Leave Protocol When a set of ℓ users $\mathcal{L} = \{U_{l_1}, U_{l_2}, \dots, U_{l_\ell}\}$ leaves the group $\mathcal{G} = \{U_1, \dots, U_n\}$, the Leave Protocol will be executed by the remaining users to change the current group key to a new group key to ensure that the leaving users cannot compute the new group key. It is assumed that controller U_1 does not leave the group. The Leave Protocol requires only 1 round of communications and is described as follows:

Controller U_1 picks a new random $r'_1 = \{1, 2, 3, \dots, \bar{N}\}$ and computes $z'_1 = g^{r'_1} \bmod \bar{N}$. Using r'_1, z'_1 and z_i for $1 \leq i \neq l_k \leq n$ where $1 \leq k \leq \ell$, U_1 then computes $y'_i = z_i^{r'_1} \bmod \bar{N}$, Y' as $Y' = \prod_{i=2, i \neq l_k}^n y'_i \bmod \bar{N}$ if n is even or as $Y' = \prod_{i=1, i \neq l_k}^n y'_i \bmod \bar{N}$ if n is odd for $1 \leq k \leq \ell$. U_1 also computes the set $T' = \{T'_i | 2 \leq i \neq l_k \leq n\}$ where $T'_i = Y' \cdot (y'_i)^{-1} \bmod \bar{N}$ for $2 \leq i \neq l_k \leq n$ and $1 \leq k \leq \ell$. Let $Z' = \{z_i | 1 \leq i \neq l_k \leq n\}$. Next, U_1 broadcast z_1, z'_1, T' and σ'_1 to the group, where σ'_1 is the signature of the broadcast message.

Upon receiving and verifying the broadcast message from U_1 , each remaining user $U_i \notin \mathcal{L}$ for $2 \leq i \neq l_k \leq n$ checks if the received z_1 is equal to the one received in the previous session and computes $Y' = (z'_1)^{r_i} \cdot T'_i \bmod \bar{N}$. All $U_i \notin \mathcal{L}$ for $1 \leq i \neq l_k \leq n$ then computes the new session key $K' = H(T' || Y')$ and updates $Z' = \{z_i | 1 \leq i \neq l_k \leq n\}$.

Join Protocol The Join Protocol is executed when a set of m new users $\mathcal{J}$ joins the group. The Join protocol will change the current group key to a new group key to ensure that the joining users are not able to compute the previous group keys that were used. It is assumed that U_1 is the controller node. The Join protocol takes two communication rounds and is described as follows:

During Round 1, each new user $U_j \in \mathcal{J}$ for $n+1 \leq j \leq n+m$ selects random $r_j = \{1, 2, 3, \dots, \bar{N}\}$ and computes $z_j = g^{r_j} \bmod \bar{N}$. $U_j \in \mathcal{J}$ broadcast z_j with signature σ_j to controller U_1 and stores its random r_j . In Round 2, after receiving and verifying all messages, U_1 selects new random $r'_1 = \{1, 2, 3, \dots, \bar{N}\}$, computes $z'_1 = g^{r'_1} \bmod \bar{N}$ and $y'_k = z_k^{r'_1} \bmod \bar{N}$ for $1 \leq k \leq n+m$. Next, U_1 computes Y' as $Y' = \prod_{k=2}^{n+m} y'_k \bmod \bar{N}$ if n is even or as $Y' = \prod_{k=1}^{n+m} y'_k \bmod \bar{N}$ if n is odd. U_1 also computes the set $T' = \{T'_k | 2 \leq k \leq n+m\}$ where $T'_k = Y' \cdot (y'_k)^{-1} \bmod \bar{N}$ for $2 \leq k \leq n+m$. Let $Z' = \{z_k | 1 \leq k \leq n+m\}$. Next, U_1 broadcast z_1, z'_1, Z', T' and σ'_1 to the group, where σ'_1 is the signature of the broadcast message.

Upon receiving and verifying the broadcast message from U_1 , users U_i for $2 \leq i \leq n$ checks to ensure that z_1 is the same as the previous round. Next, all users U_k for $2 \leq k \leq n+m$ computes

$Y' = (z'_1)^{r_k} \cdot T'_k \bmod \bar{N}$. All users U_k for $1 \leq k \leq n + m$ then computes the new session key $K' = H(T' || Y')$ and updates $Z' = \{z_k | 1 \leq k \leq n + m\}$.

2.7.9 Nam's Hierarchical Scheme

Nam et al [NKW05] proposed a hierarchical GKA protocol (Nam's Hierarchical Scheme (NHS)) that distributes n users into m subgroups and each subgroup will start the GKA protocol first to derived a shared secret. Next, each subgroup participates in the GKA protocol as a single entity to generate the final group key. The users are defined in three nonempty sets as follows: $L_0 = \{U_1\}$ which consists of main controller U_1 , $L_1 = \{U_2, \dots, U_{m+1}\}$ which consists of the subgroup controllers U_j for $2 \leq j \leq m + 1$ and $L_2 = \{U_{m+2}, \dots, U_n\}$ which consists of the users U_i for $m + 2 \leq i \leq n$. Each subgroup controller $U_j \in L_1$ for $2 \leq j \leq m + 1$ forms a subgroup with its subgroup members (see Figure 2.6) and takes charge of the central control in that subgroup. It should be noted that the tree used in NHS as shown in Figure 2.6 is not a binary tree but a generalized tree structure. Nam et al [NKW05] did not proposed any dynamic GKA protocols in their paper.

Two random large primes, q and p , are selected such that q divides $p - 1$ (p is normally 1024 bits and q is normally 160 bits). Next, an element $g \in Z_p^*$ of order q is selected. Let H be a one way cryptographic hash function where $H : \{0, 1\}^* \rightarrow \{0, 1\}^\lambda$, where λ is an integer. For simplicity, “mod p ” is omitted from the expressions if no confusion arises. I_j is denoted as the set of indices for the subgroup members of subgroup controller U_j . The NHS protocol is described as follows:

Figure 2.6: Nam's Hierarchical Scheme with n users

In Round 1, each user $U_i \in L_2$ for $m + 2 \leq i \leq n$ selects random $r_i \in Z_q$ and computes $z_i = g^{r_i}$ and sends z_i to its subgroup controller. Main controller U_1 chooses random $s_1, k_1 \in Z_q$ and computes $\omega_1 = g^{s_1}$ and $\hat{x}_1 = g^{s_1 k_1}$, and subgroup controller $U_j \in L_1$ chooses random $s_j, r_j \in Z_q$ and computes

$\omega_j = g^{s_j}$ and $x_j = g^{s_j r_j}$. Next in Round 2, each subgroup controller $U_j \in L_1$, upon receiving z_i for $i \in I_j$, computes $x_i = z_i^{s_j}$. After computing $X_j = \prod_{i \in I_j \cup \{j\}} x_i$, the set $Y_j = \{y_i | i \in I_j\}$, where $y_i = X_j \cdot x_i^{-1}$, the subgroup key $k_j = H(Y_j || X_j)$ and $\hat{z}_j = g^{k_j}$, U_j broadcasts $m_j = \hat{z}_j || \omega_j || Y_j$. In Round 3, main controller $U_1 \in L_0$, upon receiving m_j for $2 \leq j \leq m+1$, computes $\hat{x}_j = \hat{z}_j^{s_1}$. After computing $X_1 = \prod_{j \in [1, m+1]} \hat{x}_j$, $Y_1 = \{\hat{y}_j | j \in [2, m+1]\}$, where $\hat{y}_j = X_1 \cdot \hat{x}_j^{-1}$, U_1 broadcasts $\omega_1 || Y_1$. In the Key Computation round, all users U_i , subgroup controllers U_j and main controller U_1 are able to generate the session key K by first calculating $k_j = H(Y_j || X_j)$ with $X_j = y_i \cdot \omega_j^{r_i}$ and then $K = H(Y_1 || X_1)$ with $X_1 = \hat{y}_j \cdot \omega_1^{k_j}$.

2.7.10 Saeednia and Safavi-Naini scheme

Saeednia and Safavi-Naini [SSN98] proposed an identity-based GKA protocol called the SSN scheme that is based on the BD GKA protocol described in Section 2.7.6. Their idea is to use the identity information of adjacent principals to gain authentication of keying materials. A composite modulus $\bar{N} = \bar{p}\bar{q}$ is selected, where $\bar{p}$ and $\bar{q}$ are large primes such that $\bar{p} - 1 = 2\bar{p}'$, $\bar{q} - 1 = 2\bar{q}'$ and $\bar{p}'$ and $\bar{q}'$ are also primes. Next, two bases g and h of order $\bar{p}'\bar{q}'$ are chosen by the Trusted Authority (TA). A large integer $u < \bar{p}'\bar{q}'$ and one way cryptographic hash function H are also selected. The group size is denoted as n while $||$ denotes concatenation. During the initialization phase, all users are issued two secrets by the TA, namely $S_i = g^{ID_i^{-1} \bmod \bar{p}'\bar{q}'} \bmod \bar{N}$ and $T_i = h^{-ID_i^{-1} \bmod \bar{p}'\bar{q}'} \bmod \bar{N}$, where ID_i corresponds to the identity of user U_i for $1 \leq i \leq n$. A form of signature is broadcast by all users, thereby allowing stronger authentication. Their protocol consists of two phases and is described as follows:

In Phase 1, each user U_i for $1 \leq i \leq n$ chooses $r_i \in Z_u$, computes and broadcasts $t_i = g^{r_i} \bmod \bar{N}$. Upon receiving all t_j for $1 \leq j \neq i \leq n$, each U_i computes $c = H(t_1 || t_2 || \dots || t_n)$, $X_i = (t_{i+1}/t_{i-1})^{r_i} \bmod \bar{N}$ and $\omega_i = S_i^{H(X_i)r_i} T_i^c \bmod \bar{N}$. Next in Phase 2, each U_i broadcast X_i and ω_i . Upon receiving all X_j and ω_j , U_i checks $\omega_j^{ID_j} h^c \stackrel{?}{=} t_j^{H(X_j)} \bmod \bar{N}$ for $1 \leq j \neq i \leq n$. If the check is correct, U_i computes the shared group key $K = t_{i-1}^{nr_i} X_i^{n-1} X_{i+1}^{n-2} \dots X_{i-2} = g^{r_1 r_2 + r_2 r_3 + \dots + r_n r_1} \bmod \bar{N}$.

2.7.11 Kim-Lee-Lee Group Key Agreement Protocols

Kim et al [KLL04] proposed three authenticated GKA protocol, namely Initial GKA (Setup) protocol, Join and Leave Protocol, for ad hoc networks. Their protocols requires two rounds of communications and each user executes three modular exponentiations and at most $\mathcal{O}(n)$ XOR operations. A group key space belongs to $\{0, 1\}^\lambda$ where λ is an integer. Let $G = \langle g \rangle$ be a cyclic group of prime order p where g and p are public parameters and $l \leq |p|$ is satisfied. Let $H : \{0, 1\}^* \rightarrow \{0, 1\}^\lambda$ be a one-way hash function where λ is an integer, and $\sigma = \mathcal{S}(m)$ as signature of message m . Their protocols are as follows:

Initial GKA Protocol

Let $\mathcal{G} = \{U_1, \dots, U_n\}$ be an initial group, where n is the group size. Their GKA protocol considers a so-called ring structure among the members in $\mathcal{G}$, i.e. members' indices could be considered on the circulation of $\{1, \dots, n\}$. Let $I_0 = ID_1 || \dots || ID_n$. Their Initial GKA (Setup) protocol is described as follows:

Round 1: Each member U_i for $1 \leq i \leq n$ randomly chooses $\kappa_i \in \{0, 1\}^\lambda$ and $r_i \in Z_p^*$, computes $z_i = g^{r_i} \bmod p$ and keeps κ_i secretly. The last member U_n computes $H(\kappa_n || 0)$. Each member U_i for $1 \leq i \leq n - 1$ broadcasts $\{z_i || \sigma_i\}$ and last member U_n broadcasts $\{H(\kappa_n || 0) || z_n || \sigma_n\}$, where signature $\sigma_i = \mathcal{S}(z_i || I_0 || 0)$ and $\sigma_n = \mathcal{S}(H(\kappa_n || 0) || z_n || I_0 || 0)$.

Round 2: Each member U_i for $1 \leq i \leq n$ verifies the signature σ_j for $1 \leq j \neq i \leq n$. The GKA protocol fails and halts if there are invalid signatures. Otherwise, U_i computes $L_i = H(z_{i-1}^{r_i} || I_0 || 0)$, $R_i = H(z_{i+1}^{r_i} || I_0 || 0)$ and $X_i = L_i \oplus R_i$. The last member U_n additionally computes $\hat{X} = \kappa_n \oplus R_n$. Each member U_i for $1 \leq i \leq n - 1$ broadcasts $\{\kappa_i || X_i || \sigma'_i\}$ and last member U_n broadcasts $\{\hat{X} || X_n || \sigma'_n\}$, where signature $\sigma'_i = \mathcal{S}(\kappa_i || X_i || I_0 || 0)$ and $\sigma'_n = \mathcal{S}(\hat{X} || X_n || I_0 || 0)$.

Key Computation: Each member U_i for $1 \leq i \leq n - 1$ verifies σ'_j for $1 \leq j \neq i \leq n$. If all signatures are valid, U_i computes

$$\bar{R}_{i+1} = X_{i+1} \oplus R_i, \quad \bar{R}_{i+2} = X_{i+2} \oplus \bar{R}_{i+1}, \quad \dots, \quad \bar{R}_{i-1} = X_{i-1} \oplus \bar{R}_{i-2}$$

and compares $\bar{R}_{i-1}$ with L_i . If $\bar{R}_{i-1} = L_i$, then U_i has R_n and uses R_n to obtain $\bar{\kappa}_n$ from $\hat{X}$. Next,

U_i checks if $H(\bar{\kappa}_n||0) = H(\kappa_n||0)$ holds. If the equation is valid, U_i for $1 \leq i \leq n$ computes the group key K as follows:

$$K = H(\kappa_1||\kappa_2||\cdots||\kappa_{n-1}||\kappa_n||0) \quad (2.4)$$

Post Computation: Each member U_i for $1 \leq i \leq n$ computes $\alpha_i = H(z_{i-1}^{r_i}||K||0)$, $\beta_i = H(z_{i+1}^{r_i}||K||0)$ and $\delta = H(\kappa_n||K||0)$ and stores $(\alpha_i, \beta_i, \delta, K)$ in secure storage.

Join Protocol

Let $\mathcal{G} = \{U_1, \dots, U_n\}$ be the group that a set of m new members $\mathcal{J} = \{U_{n+1}, \dots, U_{n+m}\}$ wishes to join. $\mathcal{G}$ is divided into three parts $\{U_1\}$, $\{U_2, \dots, U_{n-1}\}$ and $\{U_n\}$, and U_2 is considered as an agent of $\{U_2, \dots, U_{n-1}\}$. For convenience, let U_{n+m+1} , U_{n+m+2} and U_{n+m+3} denote U_1 , U_2 and U_n respectively. Let $\mathcal{G}' = \{U_{n+1}, \dots, U_{n+m+3}\}$ and $I' = ID_1||\cdots||ID_{n+m}$. A so-called ring structure is considered among the members in $\mathcal{G}'$. Their Join Protocol is described as follows:

Round 1: Each member U_j for $n+1 \leq j \leq n+m+3$ randomly chooses $\kappa_j \in \{0,1\}^\lambda$ and $r_j \in Z_p^*$, computes $z_j = g^{r_j} \bmod p$ and keeps κ_j secretly. The member $U_{n+m+2}(= U_2)$ computes $z_{n+m+2} = g^\delta \bmod p$ and the member $U_{n+m+3}(= U_n)$ computes $H(\kappa_{n+m+3}||v)$, where index v , which starts from 0 in the Initial GKA Protocol, will increase by 1 whenever the Join or Leave Protocol is executed. Each member U_j for $n+1 \leq j \leq n+m+2$ broadcasts $\{z_j||\sigma_j\}$ and U_{n+m+3} broadcasts $\{H(\kappa_{n+m+3}||v)||z_{n+m+3}||\sigma_{n+m+3}\}$, where signature $\sigma_j = \mathcal{S}(z_j||I_v||v)$ and $\sigma_{n+m+3} = \mathcal{S}(H(\kappa_{n+m+3}||v)||z_{n+m+3}||I_v||v)$.

Round 2: Each member U_j for $n+1 \leq j \leq n+m+3$ verifies signature σ_k for $n+1 \leq k \neq j \leq n+m+3$. If there are invalid signatures, then the protocol fails and halts. Otherwise, each U_j computes $L_j = H(z_{j-1}^{r_j}||I_v||v)$, $R_j = H(z_{j+1}^{r_j}||I_v||v)$ and $X_j = L_j \oplus R_j$. The member U_{n+m+3} additionally computes $\widehat{X'} = \kappa_{n+m+3} \oplus R_{n+m+3}$. Each member U_j for $n+1 \leq j \leq n+m+2$ broadcasts $\{\kappa_j||X_j||\sigma'_j\}$ and member U_{n+m+3} broadcasts $\{\widehat{X'}||X_{n+m+3}||\sigma'_{n+m+3}\}$, where signature $\sigma'_j = \mathcal{S}(\kappa_j||X_j||I_v||v)$ and $\sigma'_{n+m+3} = \mathcal{S}(\widehat{X'}||X_{n+m+3}||I_v||v)$. Members $U_3, U_4, \dots, U_{n-1}$ compute L_{n+m+2} and R_{n+m+2} using δ .

Key Computation: Each member U_j for $n+1 \leq j \leq n+m+3$ verifies signature σ'_k for $n+1 \leq$

$k \neq j \leq n + m + 3$. If all signatures are valid, U_j computes

$$\bar{R}_{j+1} = X_{j+1} \oplus R_j, \quad \bar{R}_{j+2} = X_{j+2} \oplus \bar{R}_{j+1}, \quad \dots, \quad \bar{R}_{j-1} = X_{j-1} \oplus \bar{R}_{j-2}$$

and check if $\bar{R}_{j-1} = L_j$ holds. The members $U_3, U_4, \dots, U_{n-1}$ can check by using L_{n+m+2} and R_{n+m+2} . Finally all members U_i for $1 \leq i \leq n + m$ can obtain κ_{n+m+3} from $\widehat{X'}$ and compute new group key K' as follows:

$$K' = H(\kappa_{n+1} || \kappa_{n+2} || \dots || \kappa_{n+m+3} || v) \quad (2.5)$$

Post Computation: Each member U_j for $n+1 \leq j \leq n+m$ computes $\alpha_j = H(z_{j-1}^{r_j} || K' || v)$ and $\beta_j = H(z_{j+1}^{r_j} || K' || v)$. U_1 and U_n computes $\alpha'_1 = H(z_{n+m}^{r_1} || K' || v)$ and $\beta'_n = H(z_{n+1}^{r_n} || K' || v)$ respectively. All members U_i for $1 \leq i \leq n+m$ then compute $\delta' = H(\kappa'_n || K' || v)$, where $\kappa'_n = \kappa_{n+m+3}$. Each member U_j for $n+1 \leq j \leq n+m$ saves $(\alpha_j, \beta_j, \delta', K')$ secretly, members U_1 and U_n saves $(\alpha'_1, \beta_1, \delta', K')$ and $(\alpha_n, \beta'_n, \delta', K')$ secretly respectively and all other members U_i for $2 \leq i \leq n-1$ stores $(\alpha_i, \beta_i, \delta', K')$ in secure storage.

Leave Protocol

Suppose a set of ℓ users $\mathcal{L} = \{U_{l_1}, \dots, U_{l_\ell}\}$ with $\{l_1, \dots, l_\ell\} \subset \{1, 2, \dots, n\}$ wishes to leave group $\mathcal{G} = \{U_1, \dots, U_n\}$. Let $\mathcal{N}(\mathcal{R})$ be the set of all left and right neighbors of leaving users, i.e. $\mathcal{N}(\mathcal{R}) = \{U_{l_1-1}, U_{l_1+1}, \dots, U_{l_\ell-1}, U_{l_\ell+1}\}$, and re-index the new group $\mathcal{G}' = \mathcal{G} \setminus \mathcal{L} = \{V_1, \dots, V_{(n-\ell)}\}$. Let $I' = ID_1 || \dots || ID_{n-\ell}$. A ring structure is considered among the members in $\mathcal{G}'$. Their Leave Protocol is described as follows:

Round 1: Each member $U_j \in \mathcal{N}(\mathcal{R})$ randomly chooses $\kappa_j \in \{0, 1\}^\lambda$ and $r_j \in Z_p^*$, computes $z_j = g^{r_j} \bmod p$ and keeps κ_j secretly. The member $U_{l_\ell+1}$ computes $H(\kappa_{l_\ell+1} || v)$, where index v , which starts from 0 in the Initial GKA Protocol, will increase by 1 whenever the Join or Leave Protocol is executed. Each member U_j broadcasts $\{z_j || \sigma_j\}$ and $U_{l_\ell+1}$ broadcasts $\{H(\kappa_{l_\ell+1} || v) || z_{l_\ell+1} || \sigma_{l_\ell+1}\}$, where signature $\sigma_j = \mathcal{S}(z_j || I_v || v)$ and $\sigma_{l_\ell+1} = \mathcal{S}(H(\kappa_{l_\ell+1} || v) || z_{l_\ell+1} || I_v || v)$.

Round 2: All members $V_i \in \mathcal{G}'$ for $1 \leq i \leq n - \ell$ verify signatures σ_j and $\sigma_{l_\ell+1}$. If there are invalid signatures, then the protocol fails and halts. Otherwise, each member $U_{l_k-1} \in \mathcal{N}(\mathcal{R})$ changes

$\beta_{l_k-1} = z_{l_k+1}^{r_{l_k-1}} \bmod p$ while $U_{l_k+1} \in \mathcal{N}(\mathcal{R})$ changes $\alpha_{l_k+1} = z_{l_k-1}^{r_{l_k+1}} \bmod p$ for $1 \leq k \leq \ell$. Then each member $V_i \in \mathcal{G}'$ computes $L'_i = H(\alpha_i || I_v || v)$, $R'_i = H(\beta_i || I_v || v)$ and $X'_i = L'_i \oplus R'_i$. The member $U_{\ell+1}$ additionally computes $\widehat{X}' = \kappa_{\ell+1} \oplus R_{\ell+1}$. Each member $U_j \neq U_{\ell+1} \in \mathcal{N}(\mathcal{R})$ broadcasts $\{\kappa_j || X'_j || \sigma'_j\}$, $U_{\ell+1}$ broadcast $\{\widehat{X}' || X'_{\ell+1} || \sigma'_{\ell+1}\}$ and members $V_j \in \mathcal{G}' \setminus \mathcal{N}(\mathcal{R})$ broadcast $\{X'_j || \sigma'_j\}$, where signature $\sigma'_j = \mathcal{S}(\kappa_j || X'_j || I_v || v)$, $\sigma'_{\ell+1} = \mathcal{S}(\widehat{X}' || X'_{\ell+1} || I_v || v)$ and $\sigma'_j = \mathcal{S}(X'_j || I_v || v)$.

Key Computation: All members $V_i \in \mathcal{G}'$ for $1 \leq i \leq n - \ell$ verify the received signatures. If all signatures are valid, $V_i \in \mathcal{G}'$ computes

$$\overline{R}_{i+1} = X_{i+1} \oplus R_i, \quad \overline{R}_{i+2} = X_{i+2} \oplus \overline{R}_{i+1}, \quad \dots, \quad \overline{R}_{i-1} = X_{i-1} \oplus \overline{R}_{i-2}$$

and check if $\overline{R}_{i-1} = L_i$ holds. Then all members can obtain $\kappa_{\ell+1}$ from $\widehat{X}'$ and compute new group key K' as follows:

$$K' = H(\kappa_{l_1-1} || \kappa_{l_1+1} || \dots || \kappa_{l_\ell-1} || \kappa_{l_\ell+1} || v) \quad (2.6)$$

Post Computation: Each member $V_i \in \mathcal{G}'$ for $1 \leq i \leq n - \ell$ computes $\alpha'_i = H(\alpha_i || K' || v)$, $\beta'_i = H(\beta_i || K' || v)$ and $\delta' = H(\kappa_{\ell+1} || K' || v)$ and saves $(\alpha'_i, \beta'_i, \delta', K')$ in secure storage.

Kim's GKA protocol is proven to be secure under the Computational Diffie-Hellman (CDH) assumption and a secure signature scheme by Kim et al [KLL04]. Hence we list the following theorem:

Theorem 3 *Kim's GKA protocol is secure provided the signature scheme used is secure and the CDH problem is hard.*

We refer readers to [KLL04] for details of the security proof.

2.7.12 Dutta and Barua's Dynamic Group Key Agreement Protocol

Dutta and Barua [DB05] extended the work of Burmester and Desmedt [BD95] and proposed provably secure constant round authenticated GKA protocols in dynamic scenario, where users can join or leave the group at their desire and the session key is updated. They also provided security proofs in the standard model for their dynamic authenticated GKA, namely Join and

Leave Protocol, in their paper. Dutta and Barua's Leave protocol requires only the left and right neighbors of leaving users to change their random secret elements. In this thesis, we only cryptanalyse Dutta and Barua's Leave Protocol. Hence we omit the description of Dutta and Barua's Join Protocol and only described the Dutta and Barua's Initial GKA and Leave Protocol. Interested readers may refer to Dutta and Barua's Join Protocol in [DB05]. We describe the initial group key agreement and leave protocol as follows:

Initial Group Key Agreement Protocol

Suppose a group of users $\mathcal{G} = \{U_1, \dots, U_n\}$ wishes to establish a common session key amongst themselves by running the initial Group Key Agreement protocol. The users U_i for $1 \leq i \leq n$ participating in the protocol are on a so-called ring structure and users U_{i-1} and U_{i+1} are the respective left and right neighbors of U_i for $1 \leq i \leq n$ where $U_0 = U_n$ and $U_{n+1} = U_1$. Let p and q be two large primes such that q divides $p - 1$. The protocol is defined over a multiplicative group G , which is a subgroup of Z_p^* of order q . An element $g \in Z_p^*$ of order q is selected. Let $H : \{0, 1\}^* \rightarrow Z_q^*$ be a cryptographic hash function.

Round 1: Each U_i chooses a random $r_i \in Z_q^*$, computes $z_i = g^{r_i} \bmod p$ and sends z_i to its neighbors U_{i-1} and U_{i+1} .

Round 2: Upon receiving z_{i-1} from U_{i-1} and z_{i+1} from U_{i+1} , U_i computes left key $K_i^L = z_{i-1}^{r_i} = g^{r_{i-1}r_i} \bmod p$, right key $K_i^R = z_{i+1}^{r_i} = g^{r_{i+1}r_i} \bmod p$, $Y_i = K_i^R / K_i^L$ and broadcast Y_i to the rest of the users.

Key Computation: After receiving all the messages, U_i first computes $\bar{K}_{i+1}^R = Y_{i+1} K_i^R$ using his right key K_i^R . Next, U_i computes sequentially $\bar{K}_{i+j}^R$ for $2 \leq j \leq n - 1$ using the equation:

$$\bar{K}_{i+j}^R = Y_{i+j} \cdot \bar{K}_{i+(j-1)}^R \bmod p \quad (2.7)$$

where all subscripts above are modular n and set 0 to be n . U_i then verifies if $K_{i+(n-1)}^R = \bar{K}_{i+(n-1)}^R$ (i.e. if $K_i^L = \bar{K}_{i+(n-1)}^R$). If the verification fails, then U_i aborts the protocol. Otherwise, U_i has the

correct right keys $\bar{K}_j^R$ for $1 \leq j \leq n$. U_i computes the session key sk as follows:

$$\begin{aligned} sk &= \prod_{j=1}^n \bar{K}_j^R \bmod p \\ &= g^{r_1 r_2 + r_2 r_3 + \dots + r_n r_1} \bmod p \end{aligned} \quad (2.8)$$

In addition, U_i also computes and stores $r = H(sk)$ and also stores left and right key K_i^L and K_i^R in secure storage.

From the observation of equation (2.7), we immediately have the following lemma:

Lemma 1 *For each user U_i for $1 \leq i \leq n$, where $U_0 = U_n$ and $U_{n+1} = U_1$, user U_i is able to compute the right keys $\bar{K}_{i+j}^R = g^{r_{i+j} r_{(i+j)+1}} \bmod p$ for $1 \leq j \leq n - 1$.*

Leave Protocol

Suppose a group of m users $\mathcal{L} = \{U_{l_1}, \dots, U_{l_m}\}$ wants to leave the group $\mathcal{G} = \{U_1, \dots, U_n\}$, who has already established a session key in the previous group key agreement. Then the new group of users are $\{U_1, \dots, U_{l_1}^L\} \cup \{U_{l_1}^R, \dots, U_{l_2}^L\} \cup \dots \cup \{U_{l_m}^R, \dots, U_n\}$, where $U_{l_i}^L$ and $U_{l_i}^R$ are respectively the left and right neighbors of leaving user U_{l_i} for $1 \leq i \leq m$. Then for any leaving user U_{l_i} , $U_{l_i}^L = U_{l_i-k}$ if the k consecutive users $U_{l_i}, U_{l_i-1}, \dots, U_{l_i-(k-1)}$ are all leaving and U_{l_i-k} is not leaving the group. Similarly, $U_{l_i}^R = U_{l_i+k'}$ if the k' consecutive users $U_{l_i}, U_{l_i+1}, \dots, U_{l_i+(k'-1)}$ are all leaving and $U_{l_i+k'}$ is not leaving the group. The $n - m$ remaining users are reindexed and denoted in the new user set $\mathcal{G}' = \{V_1, \dots, V_{n-m}\}$. The left and right keys of remaining users are re-indexed and denoted by two arrays $\{\hat{K}_1^L, \dots, \hat{K}_{n-m}^L\}$ and $\{\hat{K}_1^R, \dots, \hat{K}_{n-m}^R\}$ respectively. A so-called ring structure is considered for the $n - m$ users $\mathcal{G}' = \{V_1, \dots, V_{n-m}\}$.

Round 1: Let $U_{j_L} = U_{l_i}^L$ and $U_{j_R} = U_{l_i}^R$ be the left and right neighbors of a leaving user U_{l_i} respectively, where $j_L < j_R$ for $1 \leq i \leq m$. We also let U_{j_L-k} be the left neighbor of U_{j_L} where there are $k - 1$ consecutive users $U_{j_L-1}, U_{j_L-2}, \dots, U_{j_L-(k-1)}$ leaving and U_{j_L-k} is not leaving the group. Similarly, let $U_{j_R+k'}$ be the right neighbor of U_{j_R} where there are $k' - 1$ consecutive users $U_{j_R+1}, U_{j_R+2}, \dots, U_{j_R+(k'-1)}$ leaving and $U_{j_R+k'}$ is not leaving the group.

First, the left neighbor U_{j_L} and right neighbor U_{j_R} of a leaving user U_{l_i} randomly choose new secret elements $\hat{r}_{j_L}, \hat{r}_{j_R} \in Z_q^*$ and compute $\hat{z}_{j_L} = g^{\hat{r}_{j_L}} \bmod p$ and $\hat{z}_{j_R} = g^{\hat{r}_{j_R}} \bmod p$ respectively. U_{j_L}

then sends $\hat{z}_{j_L}$ to its left neighbor U_{j_L-k} and its right neighbor U_{j_R} , while U_{j_R} sends $\hat{z}_{j_R}$ to its left neighbor U_{j_L} and its right neighbor $U_{j_R+k'}$ respectively.

Round 2:

(i) After receiving the messages from its neighbors, user U_{j_L} modifies its left key $\hat{K}_{j_L}^L = \hat{z}_{j_L-k}^{\hat{r}_{j_L}} \bmod p$ and right key $\hat{K}_{j_L}^R = \hat{z}_{j_R}^{\hat{r}_{j_L}} \bmod p$ where $\hat{z}_{j_L-k} = z_{j_L-k}$ if the left neighbor of U_{j_L-k} did not leave the group. Similarly, user U_{j_R} modifies its left key $\hat{K}_{j_R}^L = \hat{z}_{j_L}^{\hat{r}_{j_R}} \bmod p$ and right key $\hat{K}_{j_R}^R = \hat{z}_{j_R+k'}^{\hat{r}_{j_R}} \bmod p$ where $\hat{z}_{j_R+k'} = z_{j_R+k'}$ if the right neighbor of $U_{j_R+k'}$ did not leave the group.

(ii) After receiving the messages from its neighbors, U_{j_L-k} , the left neighbor of U_{j_L} , modifies its right key $\hat{K}_{j_L-k}^R = \hat{z}_{j_L}^{\hat{r}_{j_L-k}} \bmod p$ and $U_{j_R+k'}$, the right neighbor of U_{j_R} , modifies its left key $\hat{K}_{j_R+k'}^L = \hat{z}_{j_R}^{\hat{r}_{j_R+k'}} \bmod p$ respectively where $\hat{r}_{j_L-k} = r_{j_L-k}$ if the left neighbor of U_{j_L-k} did not leave the group and $\hat{r}_{j_R+k'} = r_{j_R+k'}$ if the right neighbor of $U_{j_R+k'}$ did not leave the group.

(iii) Each user $V_i \in \mathcal{G}'$ for $1 \leq i \leq n-m$ computes $\hat{Y}_i = \hat{K}_i^R / \hat{K}_i^L$ and sends $\hat{Y}_i$ to the remaining users. Note that $\hat{K}_i^R = \hat{K}_{i+1}^L$ for $1 \leq i \leq n-m-1$, $\hat{K}_{n-m}^R = \hat{K}_1^L$ and $\hat{K}_{i+(n-m-1)}^R = \hat{K}_i^L$.

Key Computation: After receiving the messages, each user $V_i \in \mathcal{G}'$ for $1 \leq i \leq n-m$ first computes $\tilde{K}_{i+1}^R = \hat{Y}_{i+1} \cdot \hat{K}_i^R$ using his own right key $\hat{K}_i^R$. Next, V_i computes sequentially $\tilde{K}_{i+j}^R$ for $2 \leq j \leq n-m-1$ using equation $\tilde{K}_{i+j}^R = \hat{Y}_{i+j} \cdot \tilde{K}_{i+(j-1)}^R \bmod p$. V_i then verifies if $\hat{K}_{i+(n-m-1)}^R = \tilde{K}_{i+(n-m-1)}^R$ (i.e. if $\hat{K}_i^L = \tilde{K}_{i+(n-m-1)}^R$). If the verification fails, then V_i aborts the protocol. Otherwise, V_i has the correct right keys $\tilde{K}_j^R$ for $1 \leq j \leq n-m$. V_i computes the session key sk' as follows:

$$sk' = \prod_{j=1}^{n-m} \tilde{K}_j^R \bmod p. \quad (2.9)$$

In addition, V_i also computes and stores $x' = H(sk')$ and also stores his left and right key $\hat{K}_i^L$ and $\hat{K}_i^R$ in secure storage.

2.7.13 Other GKA protocols in Current Literature

Besides the above-mentioned GKM protocols, there have also been other GKM protocols published in current literature. Ateniese et al [AST00] proposed two methods that extend the GDH.2 protocol to provide authenticated group key agreement. In Ateniese et al's authenticated protocol [AST00],

the first phase is similar to the GDH.2, however, in the second phase, the last member, U_n sends $K^{r_i^{-1}}\mathcal{K}_i$ where $\mathcal{K}_i$ is the secret U_n shares with each U_i . On receipt of this message, U_i can find the common group key, K by raising the received message to $r_i\mathcal{K}_i^{-1}$.

Bresson et al [BCPQ01,BCP01,BCP02a] also proposed different authenticated version of GDH.2. In [BCPQ01] and [BCP01], they defined a formal model for the authenticated group Diffie-Hellman key exchange and proved secure protocols within this model. They used an ideal hash function [BR93] in both papers, without dealing with dynamic group changes in [BCPQ01] and concurrent executions of the protocol in [BCP01]. In [BCP02a], they refined the formal model in [BCP01] and added important attributes to it. Their security proof in [BCP02a] is in the standard model. They also exhibited a security reduction with a much tighter bound than [BCP01]. Most recently, Bresson et al presented a paper [BCP07] that revisited and combined their above-mentioned protocols [BCPQ01,BCP01,BCP02a].

Pieprzyk and Li [PL00] proposed a key agreement protocol that employs Shamir's threshold secret sharing scheme [Sha79]. The idea is to use an $(n + 1, 2n)$ threshold where each member sets up its own threshold scheme. Instead of recovering each member's secret during a protocol run, an image of the secret is found. This allows the same secret sharing scheme to be used multiple times. However, a third party is required to supply the random value and this third party has to be trusted. Several other works that use different techniques for group key agreement were also proposed in [Boy97,RBD99,DMS99,TT00].

In 2003, Bresson et al [BCEP04] proposed a group key agreement protocol for low-power devices where one member performs most of the computations while the other members are only required to perform a fixed amount of computations. Boyd and Nieto [BN03] introduced a computationally asymmetric one-round group key agreement protocol that appears to easily extend to address the dynamic case. However, this protocol does not achieve forward secrecy [BCP07].

In 2004, Bresson and Catalano [BC04] presented a two round protocol that is constructed by combining the properties of ElGamal encryption scheme [Gam85] and standard secret sharing techniques [Sha79]. However, Cho et al. [CNKW05] pointed out that with the increase in the number of users, the complexity of the protocol goes beyond the capabilities of low-power mobile devices. Zhou and Ravishankar [ZR04] presented an authenticated and fault-tolerant protocol for

tree-based key agreement. Their protocol uses RSA public and private key pairs to provide source authentication. Each node in the tree has a small subgroup known as its trust set. Shamir's threshold secret sharing scheme [Sha79] is then used to distribute the function of the trusted authority across the trust sets. It should be noted that the use of a secret sharing scheme would require the presence of a trusted third party that is responsible to generate the secret share. Nam et al. [NLKW04] applied a variant of the compiler presented by Katz and Yung in [KY03] and proposed an authenticated group key exchange protocol. Like the protocol of Bresson et al. [BCEP04], their protocol has a controller that performs most of the computations. However, their authenticated protocol requires an extra round of communication. In 2005, Cho et al. [CNKW05] proposed an efficient GKA protocol for low-power mobile devices. However, this protocol requires a special user U_n called the *server* who will perform most of the computation on behalf of other users in the group. Also in 2005, Trappe et al [TWL05] proposed a resource-aware conference key establishment for heterogeneous networks. In 2006, Lee et al [LLY06] considered several collaborative key agreement and authentication protocols for dynamic peer groups while Ghosh et al [GPR⁺06] and Panja et al [PMB06] proposed group key agreement protocols for wireless sensor networks, which are different from heterogeneous networks. Also in 2006, Liao et al [LM06] proposed a Tree-based GKA framework for MANETs. Besides, most current GKA protocols do not provide key confirmation and protection against DoS attacks. Furthermore, most current GKA protocols except [DPSW06] discuss only on security against outsider and do not focus on attacks by malicious group insiders.

The IEEE 802.11i standard specifies a GKA protocol for Wireless LAN (WLAN) but not for Heterogeneous Wireless Networks (HWNs). It was also mentioned in [CJL05] that more work is needed in IEEE 802.11i to develop a more secure WLAN environment. Most recently in 2006, Hegland et al [HWM⁺06] presented a survey paper of key management in ad hoc networks but not HWNs which discuss various contributory and distributive key management protocols in current literature. Besides these, there are also other GKM protocols such as [SSKV06,DSL06] proposed in 2006. However, all the above-mentioned protocols do not provide protection against DoS attacks.

Gangwar and Sarje [GS06] briefly presented some group key agreement protocols (CLIQUES protocol suite, TGDH, STR and NAGKA) and analyze them for efficient dynamic membership events for judging their suitability for an ad hoc network. Choudhary et al [CART07] proposed

a GKA protocol for Ad Hoc Sensor Networks that adapts existing work on Tree-based Group Key Agreement that combines key trees with Diffie-Hellman key exchange, by replacing expensive public key operations with relatively cheaper symmetric key operations. Niu [Niu07] proposed a GKA protocol using ECDH (Elliptic Curve Diffie-Hellman) based on GDH.3 (Group Diffie-Hellman) protocol for Mobile Ad Hoc Networks, however Niu did not based their simulation on low-power energy constrained user device such as PDAs which are used in Mobile Ad Hoc Networks. In 2007, Yu et al [YSL07] proposed a contributory GKA protocol that optimizes rekeying cost by utilizing a novel logical key tree structure known as PFMH, and the concept of phantom user position. Tseng [Tse07] proposed a secure authenticated group key agreement protocol for resource-limited mobile nodes, however they did not provide energy analysis to analyze the energy used for computation and communication (both transmission and reception) in their GKA protocol. Lazos and Poovendran [LP07] proposed a power proximity based key management protocol for secure multicast in ad hoc networks while Wang et al [WFH07] proposed a hierarchy-based key management protocol for secure group communications in mobile ad hoc networks. Other GKA protocols [SLS07, KK07, NHSC07] have also been proposed most recently in 2007. However, they do not perform any energy analysis nor offer protection against DoS attacks.

2.7.14 Password-based GKA Protocols in Current Literature

Most current password-based key exchange protocols such as [Mac02, AP05, ACP05, MPS00, AFP06, Str06, SKI06, WZ06, YY06, YFH07, HNY07] are based on two or three party key exchange setting and not on group key agreement setting. In 2002, Bresson et al [BCP02b] proposed a password-based solution to the group Diffie-Hellman key exchange problem. However the total number of rounds in their protocol was linear to the total number of nodes. Lee et al [LHL04] and more recently in 2006, Dutta and Barua [DB06] proposed password-based GKA protocols that are based on the constant round Burmester-Desmedt [BD95] protocol. Unfortunately, both [LHL04] and [DB06] were shown to be not secure in [ABCP06]. Most recently, Dutta [Dut07] presented a simple and elegant contributory GKA protocol in password-based setting, derived from the GKA protocol in [DB05], which is a simple variant of the Burmester-Desmedt protocol [BD95]. All the above-mentioned password-based GKA protocols are based on a common password setting where all group members

share the same password. However, leaving users will still know the common password and it may not be easy to securely update every remaining group member with the new password. Furthermore, all the above-mentioned password-based GKA protocols also do not provide protection against DoS attacks and on-line password guessing attacks.

We shall now present two Password-based GKA protocols, namely Abdalla et al's Password-based GKA Protocol [ABCP06] and Byun et al's EKE-D Protocol [BLLH06] in details below:

2.7.15 Abdalla et al's Password-based GKA Protocol

Abdalla et al [ABCP06] proposed a password-based constant round GKA protocol that is based on the BD protocol. Their protocol requires only four rounds of communications and four multi-exponentiations per user. However, they did not propose any dynamic GKA protocols in their paper [ABCP06].

Two random large primes, q and p , are selected such that q divides $p-1$ (p is normally 1024 bits and q is normally 160 bits). Next, an element $g \in Z_p^*$ of order q is selected. Two cryptographic hash functions H_1 and H_2 are defined as follows: $H_1 : \{0, 1\}^* \rightarrow \{0, 1\}^{\lambda_{H_1}}$ and $H_2 : \{0, 1\}^* \rightarrow \{0, 1\}^{\lambda_{H_2}}$, where λ_{H_1} and λ_{H_2} are integers. The key confirmation uses function $Auth : \{0, 1\}^* \rightarrow \{0, 1\}^{\lambda_{Auth}}$, where λ_{Auth} is an integer. Their GKA protocol also makes use of a symmetric key encryption scheme SKE as defined in Section 2.3.3. For simplicity, “mod p ” is omitted from the expressions if no confusion arises. Their GKA protocol is described as follows:

In Round 1, each U_i selects random nonce N_i and sends U_i, N_i . Next in Round 2, the session $S = U_1 || N_1 || \dots || U_n || N_n$ is defined in which each U_i has a specific symmetric key $k_i = H_1(S, i, pw)$, where pw is the password. Each U_i then selects random $r_i \in Z_q^*$ and broadcast $z_i^* = \mathcal{E}_{k_i}(z_i)$ where $z_i = g^{r_i}$. Next in Round 3, each U_i extracts $z_{i-1} = \mathcal{D}_{k_{i-1}}(z_{i-1}^*)$ and $z_{i+1} = \mathcal{D}_{k_{i+1}}(z_{i+1}^*)$ and computes $Z_i = z_{i-1}^{r_i}$ and $Z_{i+1} = z_{i+1}^{r_i}$. U_i then broadcasts $X_i = Z_{i+1}/Z_i$. In Round 4, each U_i computes $K_i = Z_i^n X_i^{n-1} X_{i+1}^{n-2} \dots X_{i+n-2}$, and broadcasts its key confirmation $Auth_i = Auth(S, \{z_j^*, X_j\}_j, K_i, i)$. Finally in the Key Computation Round, each U_i after receiving and checking all the key confirmations, derives the session key $sk_i = H_2(S, \{z_j^*, X_j, Auth_j\}_j, K_i)$.

2.7.16 Byun et al's EKE-D Protocol

Byun et al proposed a password-based GKA protocol known as the N-Party EKE-D protocol [BLLH06] for multi-layer MANET using different passwords, i.e. each user U_i for $1 \leq i \leq n$ has a different password pw_i that is shared with a single server/controller, where n is the group size. Their paper focus on a type of military MANET called the Unmanned Aerial Vehicle-Mobile Backbone Nodes (UAV-MBN) [CL06] network. The single server/controller node (C) is known as the Mobile Backbone Node while the user nodes are known as Group Nodes (U_i) in their protocol. The proposed a Setup protocol for group formation and two dynamic GKA protocols, namely Join and Leave Protocol for user Join and user Leave events.

Two random large primes, q and p , are selected such that q divides $p-1$ (p is normally 1024 bits and q is normally 160 bits). Next, an element $g \in Z_p^*$ of order q is selected. Cryptographic hash functions H_1, H_2, H_3 and F are defined as follows: $H_1 : \{0, 1\}^* \rightarrow \{0, 1\}^\lambda$, $H_2 : \{0, 1\}^* \rightarrow \{0, 1\}^\lambda$, $H_3 : \{0, 1\}^* \rightarrow \{0, 1\}^\lambda$ and $F : \{0, 1\}^* \rightarrow \{0, 1\}^\lambda$, where λ is an integer. The EKE-D protocol also uses a symmetric key encryption scheme SKE as defined in Section 2.3.3. For simplicity, “mod p ” is omitted from the expressions if no confusion arises. For an initial group $\mathcal{G} = \{U_1, \dots, U_n\}$, the Initial GKA (Setup) protocol is described as follows:

EKE-D Initial GKA Protocol In Round 1, the single server/controller C selects randoms $s_i \in Z_q^*$ and sends $\mathcal{E}_{pw_i}(g^{s_i})$, that is, g^{s_i} encrypted using password pw_i , to all nodes U_i for $1 \leq i \leq n$, where pw_i is the password for user U_i . Simultaneously, each U_i selects random $r_i \in Z_q^*$ and sends $\mathcal{E}_{pw_i}(g^{r_i})$ to C . At the end of Round 1, C and U_i compute the ephemeral Diffie-Hellman key $t_i = H_1(\text{sid} || g^{r_i s_i})$ and $\text{sid} = \mathcal{E}_{pw_1}(g^{r_1}) || \mathcal{E}_{pw_2}(g^{r_2}) || \dots || \mathcal{E}_{pw_n}(g^{r_n})$. C then computes $\bar{t}_i = F(t_i)$ and saves a set $\text{LIST}_{sc} = \{\bar{t}_1, \dots, \bar{t}_n\}$. Next in Round 2, C first selects random $N \in Z_q^*$, computes and sends $N \oplus \bar{t}_i$ where $\oplus$ denotes XOR operation and authenticators $H_2(\bar{t}_i || C)$ to U_i for $1 \leq i \leq n$. Concurrently, U_i computes and sends authenticator $H_2(\bar{t}_i || U_i)$ to C . Then in the Key Computation round, all nodes first checks the authenticators using $\bar{t}_i$. Next, all U_i obtains N using its $\bar{t}_i$ and generates group session key $sk = H_3(\text{SIDS} || N)$ where $\text{SIDS} = \text{sid} || t_1 \oplus N || t_2 \oplus N || \dots || t_n \oplus N$ and $\text{sid} = \mathcal{E}_{pw_1}(g^{r_1}) || \mathcal{E}_{pw_2}(g^{r_2}) || \dots || \mathcal{E}_{pw_n}(g^{r_n})$. All nodes U_i then computes $\bar{t}_i = F(t_i)$ and save it in the secure storage.

Next, we describe the EKE-D Join Protocol that will be executed when a group of new users

$\mathcal{J} = \{u_{n+1}, \dots, u_{n+m}\}$ wishes to join the group $\mathcal{G} = \{u_1, \dots, u_n\}$. The EKE-D Join Protocol will change the current group key to a new group key to ensure that the joining users are not able to compute the previous group keys used. The EKE-D Join Protocol will also updates $\mathcal{G}' = \mathcal{G} \cup \mathcal{J}$. The EKE-D Join Protocol is described as follows:

EKE-D Join Protocol In Round 1, the Join Protocol will execute Round 1 of the Setup Protocol with the new user $\mathcal{J} = \{u_{n+1}, \dots, u_{n+m}\}$. $\mathcal{C}$ gets the new set $\text{LIST}_{\mathcal{J}} = \{\bar{t}_{n+1}, \dots, \bar{t}_{n+m}\}$ which consists of intermediate DH keys $\bar{t}_j$ for $n+1 \leq j \leq n+m$. $\mathcal{C}$ then updates $\text{LIST}'_{\text{sc}} = \text{LIST}_{\text{sc}} \cup \text{LIST}_{\mathcal{J}}$. In the second round, $\mathcal{C}$ selects new random $N' \in Z_q^*$ and computes $N' \oplus \bar{t}_i$ using $\text{LIST}'_{\text{sc}} = \{\bar{t}_1, \dots, \bar{t}_{n+m}\}$. $\mathcal{C}$ then broadcast $N' \oplus \bar{t}_i$ and authenticators $H_2(\bar{t}_i || \mathcal{C})$ to u_i for $1 \leq i \leq n+m$. Concurrently, u_i computes and sends authenticator $H_2(\bar{t}_i || u_i)$ to $\mathcal{C}$. Finally in the Key Computation Round, all nodes check if the authenticator is valid by using $\bar{t}_i$. Next, all nodes obtains N' using $\bar{t}_i$ and generates the new key $sk' = H_3(\text{SIDS}' || N')$ where $\text{SIDS}' = \text{sid}' || t_1 \oplus N' || t_2 \oplus N' || \dots || t_{n+m} \oplus N'$ and $\text{sid}' = \mathcal{E}_{pw_1}(g^{r_1}) || \mathcal{E}_{pw_2}(g^{r_2}) || \dots || \mathcal{E}_{pw_{n+m}}(g^{r_{n+m}})$.

Next, we describe the EKE-D Leave Protocol that will be executed when there are users leaving the group. Suppose a set of ℓ leaving users $\mathcal{L} = \{u_{l_1}, \dots, u_{l_\ell}\}$ wishes to leave the group $\mathcal{G} = \{u_1, \dots, u_n\}$. The EKE-D Leave Protocol will change the current group key to a new group key to ensure that the leaving users cannot compute the new group key. The EKE-D Leave Protocol also updates $\mathcal{G}' = \mathcal{G} \setminus \mathcal{L}$. The remaining users are reindexed and denoted as $\mathcal{G}' = \{v_1, \dots, v_{n-\ell}\}$. The EKE-D Leave Protocol is described as follows:

EKE-D Leave Protocol The Leave Protocol will invoke Round 1, Round 2 and Key Computation round of the EKE-D Initial GKA (Setup) Protocol for the remaining users v_i for $1 \leq i \leq n-\ell$. Each v_i for $1 \leq i \leq n-\ell$ will then compute the new session key $sk' = H_3(\text{SIDS}' || N')$ where $N' \in Z_q^*$, $\text{SIDS}' = \text{sid}' || t'_1 \oplus N' || t'_2 \oplus N' || \dots || t'_{n-\ell} \oplus N'$ and $\text{sid}' = \mathcal{E}_{pw_1}(g^{r'_1}) || \mathcal{E}_{pw_2}(g^{r'_2}) || \dots || \mathcal{E}_{pw_{n-m+1}}(g^{r'_{n-m+1}})$.

2.8 Summary on the General Idea of GKA Protocols

The GKA protocols surveyed in Section 2.7 can be generally classified into two different categories, namely non-controller based and controller based. In this section, we provide a summary on the general idea of GKA protocols (i.e. non-controller based and controller based GKA protocols) that

provides the basis for the works in subsequent chapters of this thesis. We also illustrate each general method with an example and discuss the strengths and limitations of each method.

2.8.1 Non-controller based GKA Protocols

Non-controller based GKA protocols are characterized by the lack of a central controller in the group. Examples of non-controller based GKA protocols are TGDH protocol [KPT00, KPT04b], STR protocol [KPT01, KPT04a], the BD protocol [BD95], SSN protocol [SSN98], Kim-Lee-Lee GKA protocol [KLL04], Dutta and Barua's GKA protocol [DB05] and Abdalla et al's password-based GKA protocol [ABCP06]. The general idea of non-controller based GKA protocols is as follows:

Phase 1: Each user node will select its random secret, compute the keying material based on the random secret and broadcast the keying material to other users. In the next round, each user would compute new keying materials based on the received keying materials and broadcast the new keying materials to other users. The number of rounds in Phase 1 depends on the non-controller based GKA protocol. For example, the BD protocol requires two rounds for Phase 1.

Phase 2: Each user would compute the final group key after receiving the necessary keying materials at the end of Phase 1 and the group key can then be used for secure group communications.

Example of non-controller based GKA protocol - The BD Protocol: We illustrate the general method of non-controller based GKA protocol with the BD protocol as described in Section 2.7.6 as follows:

Phase 1: Phase 1 corresponds to the first two rounds of the BD protocol. In Round 1 of the BD protocol, each user U_i for $1 \leq i \leq n$ selects random secret r_i , computes and broadcasts keying material $z_i = g^{r_i} \bmod p$. Then in Round 2 of the BD protocol, each U_i computes and broadcast new keying material $X_i = \left(\frac{z_{i+1}}{z_{i-1}}\right)^{r_i}$ using the received key materials z_{i+1} and z_{i-1} .

Phase 2: Phase 2 corresponds to the key computation round in the BD protocol. In this case, all users in BD can compute the group key $K = (z_{i-1}^{r_i})^n \cdot X_i^{n-1} \cdot X_{i+1}^{n-2} \cdots X_{i-2} = g^{r_1 r_2 + r_2 r_3 + \dots + r_n r_1} \bmod p$ and use K for secure group communications.

Strengths One advantage of non-controller based GKA protocols is that there is no single point of failure. Non-controller based GKA protocols are also fair to all nodes, i.e. all nodes in non-controller based GKA protocols would usually have similar computational and communication costs.

Weaknesses The computational and communication costs incurred by user nodes would be higher than controller-based GKA protocols as there is no controller node to perform most of the computation and communication on behalf of the user nodes. The handling of user join and leave in non-controller based GKA protocols would also be less efficient due to lack of central controller.

2.8.2 Controller based GKA Protocols

Controller based GKA protocols are characterized by the presence of a controller node that helps remove some of the computational and communication burdens of the user nodes. Examples of controller based GKA protocols includes GDH protocol [STW96, STW98], Bresson's GKA protocols [BCPQ01, BCP01, BCP02a], NLKW protocol [NLKW05], NLKKW protocol [NLK⁺04], NHS [NKW05] and EKE-D protocol [BLLH06]. The general idea of controller based GKA protocols is as follows:

Phase 1: Each user node will select its random secret, compute the keying material based on the random secret and broadcast the keying material to the controller node. In GKA protocols such as GDH and Bresson's GKA protocols where the nodes are arranged in a bus topology (see Section 2.9.1), then each user node U_i will send its keying material to the next user node on its right, i.e. U_i sends to U_{i+1} . U_{i+1} would then select its own random secret and compute the keying material based on the received keying material from U_i and its random secret. Next, U_{i+1} would send its keying material to U_{i+2} , who will repeat the same process as U_{i+1} . This process continues until the keying material has reached the controller node which is the last user U_n in the group.

Phase 2: The controller node would first select its own random secret and compute its own keying material. Next, the controller node would compute the group key using the keying material from all user nodes as well as its own keying material and random secret. Finally, the controller node would broadcast keying materials to all user nodes to compute the group key.

Phase 3: After receiving the keying materials sent by the controller node at the end of Phase 2, each user node would then be able to compute the group key. The group key can then be used for secure group communications.

Example of controller based GKA protocol - The NLKW Protocol: We illustrate the general method of controller based GKA protocol with the NLKW protocol as described in Section 2.7.7 as follows:

Phase 1: Phase 1 corresponds to Round 1 of the NLKW protocol where each user U_i for $2 \leq i \leq n$ selects random secret r_i , computes and broadcasts keying material $z_i = g^{r_i} \bmod p$ to controller node U_1 .

Phase 2: Phase 2 corresponds to Round 2 of the NLKW protocol. Controller node U_1 first selects its own random secrets $r, r_1 \in Z_q^*$ and compute its own keying material $z = g^r \bmod p$ and $y_1 = g^{rr_1} \bmod p$. Next, the controller node would compute the group key $K = H(T||Y)$ where $T = \{T_i | 2 \leq i \leq n\}$, $Y = \prod_{i=1}^n y_i \bmod p$, $T_i = Y \cdot y_i^{-1} \bmod p$ and $y_i = z_i^{r_i} \bmod p$. Finally, controller node U_1 broadcast keying materials z and $T = \{T_i | 2 \leq i \leq n\}$ to all user nodes to compute the group key.

Phase 3: Phase 3 corresponds to the key computation round in the NLKW protocol. After receiving the keying materials sent by the controller node at the end of Phase 2, each user U_i is able to compute the group key $K = H(T||Y)$ by first computing $Y = z^{r_i} \cdot T_i \bmod p$. The group key K can then be used for secure group communications.

Strengths One advantage of controller based GKA protocols is that the controller node would relieve most of the computational and communication burdens of the user nodes, thus improving computational and communication efficiency of user nodes. The presence of the controller node also helps improve the handling of user join and leave events.

Weaknesses The controller node may be a single point of failure in the GKA protocol. The controller node has to be a powerful device as it has to perform more computation and communication than the other user nodes. In networks where all user nodes are low power devices, then controller based GKA protocols may not be appropriate as they are unfair to the controller node.

2.9 Summary of GKA Protocol Topologies

Figure 2.7: Bus Topology

Figure 2.8: Tree Topology

Figure 2.9: Ring Topology

Figure 2.10: Star Topology

In this section, we summarize the general topology used in the various GKA protocols described in Section 2.7. The four topologies are Bus topology (Figure 2.7), Tree topology (Figure 2.8), Ring topology (Figure 2.9) and Star topology (Figure 2.10).

2.9.1 Bus topology

The bus topology (Figure 2.7) is characterized by a straight line where each node sends unicast message to its immediate right neighbor until it reaches the last user in the group. Examples of GKA protocols using the bus topology are the ING protocol [ITW82], GDH protocol [STW96, STW98] and Bresson's GKA protocols [BCPQ01, BCP01, BCP02a]. The Hypercube and Octopus protocol [BW98] uses the so-called hypercube topology (see Figure 2.2), which is a special case of the bus topology.

Advantages: The bus topology is suitable for unicast environment, especially wired networks.

Limitations: The number of rounds for GKA protocols using bus topology depends on the number of nodes in the group. Another disadvantage is that the unicast nature of bus topologies does not take full advantage of the broadcast nature of wireless networks.

2.9.2 Tree Topology

Figure 2.8 shows an illustration of the tree topology where the group key is formed at the top of the tree. The TGDH protocol [KPT00, KPT04b] and STR protocol [KPT01, KPT04a] are examples of GKA protocols that uses the tree topology.

Advantages: The tree topology improves scalability as the computation complexity increases with the height of the tree instead of the increasing with group size.

Limitations: One disadvantage is that there is a need for nodes to maintain a tree structure and update the tree structure whenever users join and leave the group.

2.9.3 Ring Topology

Figure 2.9 shows an illustration of the ring topology where nodes are arranged in a ring and broadcast messages to other nodes during the GKA protocol. Examples of GKA protocols using the ring topology are the BD protocol [BD95], SSN protocol [SSN98], Kim-Lee-Lee GKA protocol [KLL04], Dutta and Barua's GKA protocol [DB05] and Abdalla et al's password-based GKA protocol [ABCP06].

Advantages: Ring topology uses broadcast for communications, which is suitable for wireless networks. Furthermore, there is no single point of failure because there is usually no controller node in the ring topology.

Limitations: Not efficient in handling user join and user leave events.

2.9.4 Star Topology

Figure 2.10 illustrates the star topology which is characterized by the presence of a controller node that eases the computational and communication burden of user nodes by performing most of the computation and communication workload. Examples of GKA protocols using the star topology includes NLKW protocol [NLKW05], NLKKW protocol [NLK⁺04], NHS [NKW05] and EKE-D protocol [BLLH06].

Advantages: Presence of controller node improves the computation and communication efficiency of user nodes. The controller node can also handle user join and leave events more efficiently.

Limitations: Controller node may be a single point of failure.

2.10 Attacks on Group Key Management Protocols

Group key management (GKM) protocols in wireless networks are vulnerable to several attacks due to the nature of the transmission medium in wireless networks. In this section, we present some of the security attacks that targets GKM protocols in wireless networks.

Eavesdropping attack As wireless networks uses radio frequency (wireless) signals that can propagate through walls and ceilings, hence outsiders can eavesdrop on the communication data sent and received by wireless nodes. Encryption is used to provide confidentiality and protect data from eavesdropping attacks. In authentication and key exchange protocols, besides communication data, even the identities of the wireless nodes need to be protected through the use of encryption.

Replay attack Replay attacks includes any situation where the adversary interferes with a GKM protocol by insertion of a message, or part of a message, that has been previously sent in another GKM protocol run. The common countermeasure against replay attacks is through the use of nonces and timestamps to ensure that the session key generation is fresh for every protocol run.

Man-in-the-middle attack The goal of this attack is for the adversary to obtain valuable information from group members and make them believe that they are communicating with legitimate members when they are in fact communicating with the adversary. The adversary's device pretends to be a legitimate node to the group members on both sides and tries to convince the group members to share valuable information with it. If the attack is successful, the adversary would be able to know the group key used. This attack can be prevented by means of mutual authentication.

Cryptanalysis This attack is possible if the protocol is weak or if the key generated is weak and is relatively easy to guess once sufficient evidence is available. Examples of 'evidence' are pairs of information such as a plaintext and its corresponding ciphertext or a plaintext and its corresponding message authentication code. A common example of weak keys is when the key is formed from a password that is easily remembered by human. In this situation, the attacker can guess the set of values that are practically used for passwords as the effective key length in this case would be much smaller than the acceptable key length in modern cryptosystem.

Denial-of-Service (DoS) Attack GKM protocols for wireless networks are vulnerable to DoS attacks as DoS attacks can be easily launched by the adversary to disrupt GKM services and halt secure group communications. However, most GKM protocols in current literature do not discuss DoS attacks. We examined the problem of DoS attacks against GKM protocols in wireless

networks in this thesis, and proposed GKM protocols that provide protection against DoS attacks. We discuss further on the current literature in DoS attacks and countermeasures in Section 2.11.

2.11 Denial-of-Service Attacks and Countermeasures

In a Denial-of-Service (DoS) attack against GKA protocols, the adversary prevents legitimate users from completing the protocol and establishing a group key, thus preventing secure group communications from taking place. DoS attacks are usually targeted against servers/controllers who are required to interact with the user nodes. Examples of DoS attacks include depleting the computational resources of the server/controller, therefore making the server/controller too busy to accept new connections and depleting the storage resources of the server/controller. To date, there have been several measures proposed to reduce and prevent the impact of DoS attacks. As GKA protocols for wireless networks are also vulnerable to DoS attacks, therefore it is important not to ignore this issue. We discuss the current literature on DoS attack prevention research below:

1. Aura and Nikander [AN97] proposed using stateless connections to protect against DoS attacks in wired network. Their idea is to make the client store all the state information required by the server and return it to the server as necessary with each message sent. In this way, the server does not have to store the state information. However, it is necessary for the state information returned to the server to be authenticated and this information may also have to be confidential. Therefore, there is an overhead in both communication and computation incurred by transforming protocols to provide stateless connections. Karn and Simpson [KS99] suggested a way for delaying the need for state at the server through the use of cookies. Their idea is for the server to send a cookie to the client when the client attempts to make a connection. This procedure is similar to the use of cookies by the World Wide Web servers but the cookies in [KS99] are a function of a secret known only to the server and other information unique to the particular connection. The server does not need to store the connection state information for this request while the client has to return the cookie in the next message and its validity checked by the server from the information sent and its secret. The idea is to ensure that before investing significant resources, the server is certain that the

client is making a genuine request for connection. This technique prevents the DoS attacks whereby the attacker sends random connection requests. By changing the server secret on a regular basis (e.g. every 1 minute), even the same client cannot make unlimited connection request.

2. Meadows [Mea99] suggested that all messages in a protocol must be authenticated in order to prevent DoS attacks. The use of cookies as mentioned above may be a form of weak authentication to minimize the computation and communication overheads. Meadows designed a formal framework based on the idea of fail-stop protocols introduced by Gong and Syverson [GS95] which abort as soon as a bogus message is discovered.
3. Juels and Brainard [JB99] proposed a mechanism that all client puzzles to form stronger authentication than that provided by cookies. Their idea is that when load on a server becomes high, the server will send a ‘puzzle’ of moderate computation difficulty to each client which must be solved before a new connection can be made. Genuine clients will only be mildly inconvenienced by this demand but an adversary trying to make multiple connections will have to solve many puzzles. Waters et al [WJHF04] further worked on this idea of client puzzle. Fung and Lee [FL02] also proposed an improved sophisticated client puzzle that is integrated with a public-key based authentication and key establishment protocol to provide resistant against DoS attacks. Lee and Fung [LF03] then extended the work by Fung and Lee [FL02] and proposed a solution against DoS attacks in public-key based authentication and key establishment using client puzzles. However, both [FL02] and [LF03] are based on two-party key setting and not group setting. In 2005, Miao [Ma05] proposed to mitigate DoS attacks using password puzzles which is a different type of client puzzle while Choi [Cho05] proposed a DoS resistant multicast authentication protocol using a new mechanism based on predictions and one-way key chains. Tseng [Tse05] proposed an authenticated key agreement protocol resistant to DoS attacks, however they concentrated only on the two-party key agreement setting and not group key agreement.

2.12 Summary of Hierarchical Group Model Topologies

In this section, we summarize the four hierarchical group models/structures used in this thesis, namely the Generalized Circular Hierarchical (C-H) group model (used in Chapter 7), the Contributory-Ring-Centralized (ContRi-Central) group model (used in Chapter 8), the Centralized-Contributory Ring (Central-ContRi) group model (used in Chapter 9) and the Centralized-Centralized (Central-Central) group model (used in Chapter 10).

2.12.1 Generalized Circular Hierarchical (C-H) Group Model

Figure 2.11: An illustration of the (C-H) Group Model with $c = 10$.

Figure 2.11 shows the proposed circular hierarchical (C-H) group model that can be used for secure group communications in a large-scale ad hoc network containing n users. Each subgroup is arranged in a circle and contains c members. There are h layers in the (C-H) group model such that there are c^v subgroups in each layer $\mathcal{L}_v$ for $0 \leq v \leq h - 1$. The group size is defined as $n = c^h$. The highest layer is denoted as $\mathcal{L}_0$ while the lowest layer is $\mathcal{L}_{h-1}$. The example in Figure 2.11 illustrates our (C-H) group model with $c = 10$. For example, if $n = 1000$, then we can have $c = 10$ and $h = 3$. When n is not a power of c , i.e. $n < c^h$, a group key can still be established easily. In this case, the lowest layer $\mathcal{L}_{h-1}$ will have either less than c^{h-1} subgroups or if there are c^{h-1} subgroups, then the last subgroup will have less than c members. It should be noted that in practice, the number of nodes in each subgroup at different layers can be different depending on the application requirements. Without loss of generality, we will assume in this thesis that the network is balanced

with every subgroup containing the same number of users, i.e. there are c nodes in each subgroup.

The structure of our (C-H) group model, as shown in Figure 2.11, looks similar to the tree based circle structure proposed by Jho et al. in [JCKY05]. However, in that paper, the authors used the structure for broadcast encryption purposes while our work is on contributory GKA. The (C-H) group model shown in Figure 2.11 also looks similar to the model used in PNNI (Private Network-to-Network Interface) [For02] for ATM networks. However, PNNI for ATM networks focuses on network management issues such as discovery of ATM networks whereas our (C-H) group model is applied on contributory GKA.

The user nodes belong to subgroups at the lowest layer $\mathcal{L}_{h-1}$. The first member of each subgroup at layer $\mathcal{L}_v$ for $1 \leq v \leq h-1$ is the subgroup controller and is also a member of a subgroup at the next higher layer $\mathcal{L}_{v-1}$, that is, the subgroup members of layers $\mathcal{L}_v$ for $0 \leq v \leq h-2$ are subgroup controllers and the first member of subgroups at the next lower layer $\mathcal{L}_{v+1}$.

Advantages: The advantages of the (C-H) Group Model are that all nodes on the same layer would have similar computational and communication costs. This makes the (C-H) Group model especially useful in large scale ad hoc networks such as Wireless Sensor Networks (WSNs) where all nodes are low-power energy-constrained devices and there are no high power devices.

Limitations: The limitation of the (C-H) Group Model is that it may not be able to efficiently handle dynamic group membership events such as user join and user leave. This is because there is no centralized group controller in each subgroup to handle the dynamic events efficiently.

The detail description of the notations for the (C-H) Group Model is given below.

Notations for the (C-H) Group Model

There are c^h users in the group and we denote the users as U_i for $0 \leq i \leq c^h - 1$. These users U_i belong to subgroups $\mathcal{G}_{h-1,j}$ at the lowest layer $\mathcal{L}_{h-1}$ for $0 \leq j \leq c^{h-1} - 1$. The subgroups in each layer $\mathcal{L}_v$ are denoted as $\mathcal{G}_{v,j}$ for $0 \leq v \leq h-1$ and $0 \leq j \leq c^v - 1$ and each subgroup has the following users:

$$\mathcal{G}_{v,j} = \{U_{jc^{h-v}}, U_{jc^{h-v}+c^{h-v-1}}, U_{jc^{h-v}+2c^{h-v-1}}, \dots, U_{jc^{h-v}+(c-1)c^{h-v-1}}\}$$

where the first member $U_{jc^{h-v}}$ is the subgroup controller of $\mathcal{G}_{v,j}$. The user $U_{jc^{h-v}+kc^{h-v-1}}$ for

$0 \leq j \leq c^v - 1$ and $0 \leq k \leq c - 1$ of subgroup $\mathcal{G}_{v,j}$ for $0 \leq v \leq h - 2$ is the subgroup controller and first member of subgroup $\mathcal{G}_{v+1,jc+k}$ at layer $\mathcal{L}_{v+1}$ where

$$\mathcal{G}_{v+1,jc+k} = \{U_{(jc+k)c^{h-v-1}}, U_{(jc+k)c^{h-v-1}+c^{h-v-2}}, U_{(jc+k)c^{h-v-1}+2c^{h-v-2}}, \dots, U_{(jc+k)c^{h-v-1}+(c-1)c^{h-v-2}}\}$$

The first member of each subgroup $\mathcal{G}_{v,k}$ for $1 \leq v \leq h - 1$ and $0 \leq k \leq c^v - 1$ at layer $\mathcal{L}_v$ forms a subgroup $\mathcal{G}_{v-1,j}$ at layer $\mathcal{L}_{v-1}$ where $j = \lfloor \frac{k}{c} \rfloor$ as follows:

$$\mathcal{G}_{v-1,j} = \{U_{jc^{h-v+1}}, U_{jc^{h-v+1}+c^{h-v}}, U_{jc^{h-v+1}+2c^{h-v}}, \dots, U_{jc^{h-v+1}+(c-1)c^{h-v}}\}$$

For any user U_i for $0 \leq i \leq c^h - 1$, we can find out the subgroup $\mathcal{G}_{h-1,j}$ that user U_i belongs to by computing $j = \lfloor \frac{i}{c} \rfloor$, where $\lfloor a \rfloor$ is the largest integer $\leq a$ (e.g. $\lfloor 3.6 \rfloor = 3$).

Figure 2.12: User nodes in the (C-H) Group Model with $c = 3$ and $h = 3$.

To illustrate the user nodes in the (C-H) Group Model, we present Figure 2.12 where there are 27 users in the network. The number of users in each subgroup is $c = 3$ and the height of the group model is $h = 3$. As shown in Figure 2.12, some user nodes may belong to more than one subgroups. For example, User 12 (U_{12}) belongs to both subgroup $\mathcal{G}_{1,1}$ at Layer $\mathcal{L}_1$ and subgroup $\mathcal{G}_{2,4}$ at Layer $\mathcal{L}_2$ while User 9 (U_9) belongs to subgroup $\mathcal{G}_{0,0}$, $\mathcal{G}_{1,1}$ and $\mathcal{G}_{2,3}$ at Layers $\mathcal{L}_0$, $\mathcal{L}_1$ and $\mathcal{L}_2$ respectively.

2.12.2 Contributory Ring-Centralized Group Model

Figure 2.13: A ContRi-Central Group Model.

Figure 2.13 shows the proposed Contributory Ring-Centralized (ContRi-Central) group model that can be used for secure group communications in a large-scale static asymmetric HWN containing n users. Depending on the application, each subgroup in our group model can be either single layer such as Subgroups 1 and 4 or multi-layer such as Subgroups 2 and 3 in Figure 2.13. For the ContRi-Central group model described in this thesis, we shall focus on the multi-layer structure as shown in Subgroups 2 and 3 in Figure 2.13. It should be noted that the ContRi-Central group model shown in Figure 2.13 is different from the (C-H) Group Model described in Section 2.12.1. In the multi-layer ContRi-Central group model, the lowest layer $\mathcal{L}_{h-1}$ consists of all the user nodes while the next higher layer $\mathcal{L}_{h-2}$ consist of a set of subgroup controllers that control subgroups at layer $\mathcal{L}_{h-1}$. The subgroup controllers at layer $\mathcal{L}_{h-2}$ are not related to the users at layer $\mathcal{L}_{h-1}$. The next higher layer $\mathcal{L}_{h-3}$ is another set of subgroup controllers that control subgroups in layer $\mathcal{L}_{h-2}$ and the set of subgroup controllers at $\mathcal{L}_{h-3}$ are not related to the subgroup controllers at $\mathcal{L}_{h-2}$. This continues until the highest layer $\mathcal{L}_0$. The subgroup at the highest layer $\mathcal{L}_0$ is denoted as the Control Group (CG) and there are no subgroup controller for CG.

The number of nodes in the various subgroups at the various layers can vary depending on the number of high power and low power nodes in the underlying application. Without loss of generality, we assume that the network is balanced with every subgroup containing the same number of users, i.e. there are c nodes in each subgroup.

Advantages: The advantages of the ContRi-Central group model is that the subgroup controllers (high power nodes) in each of the higher layers can relieve the computational workload of the user nodes (low power nodes) at the lowest layer by performing most of the computation to obtain the group key. This group model is also useful for large scale wireless networks where there is no need for an overall main controller in the Control Group, e.g. in a mesh network where the Control Group comprises of a group of mesh routers.

Limitations: One limitation of the ContRi-Central group model is that the subgroup controller may be a single point of failure in each of the subgroup. Countermeasures include having tamper resistant protection, backup subgroup controllers and rotation off subgroup controller roles.

The detail description of the notations for the ContRi-Central Group Model is given below.

Notations for the ContRi-Central Group Model

For layer $\mathcal{L}_v$ for $0 \leq v \leq h-2$, the subgroup controllers are denoted as $\mathbf{S}_{v,j}$ for $0 \leq j \leq c^{v+1} - 1$. A subgroup controller $\mathbf{S}_{v,j}$ controls subgroup $\mathcal{G}_{v+1,j} = \{\mathbf{S}_{v+1,jc}, \mathbf{S}_{v+1,jc+1}, \mathbf{S}_{v+1,jc+2}, \dots, \mathbf{S}_{v+1,jc+c-1}\}$. The user nodes are represented at the lowest layer $\mathcal{L}_{h-1}$ as $\mathbf{S}_{h-1,i}$ for $0 \leq i \leq c^h - 1$. For layers $\mathcal{L}_v$ for $1 \leq v \leq h-1$, we denote the subgroup as $\mathcal{G}_{v,j}$ for $0 \leq j \leq c^v - 1$ and its subgroup members as $\mathbf{S}_{v,jc+k}$ for $0 \leq k \leq c-1$, that is, $\mathcal{G}_{v,j} = \{\mathbf{S}_{v,jc}, \mathbf{S}_{v,jc+1}, \mathbf{S}_{v,jc+2}, \dots, \mathbf{S}_{v,jc+c-1}\}$.

The control group (CG) at the highest layer $\mathcal{L}_0$ is denoted as $\mathcal{G}_{0,0} = \{\mathbf{S}_{0,0}, \mathbf{S}_{0,1}, \mathbf{S}_{0,2}, \dots, \mathbf{S}_{0,c-1}\}$ and $\mathcal{G}_{0,0}$ does not have subgroup controllers, therefore $\mathbf{S}_{-1,0}$ is null (empty).

For each node $\mathbf{S}_{v,k}$ at layer $\mathcal{L}_v$ for $1 \leq v \leq h-1$ and $0 \leq k \leq c^{v+1} - 1$, we can find its subgroup controller $\mathbf{S}_{v-1,j}$ by computing $j = \lfloor \frac{k}{c} \rfloor$. We can also find its (super) subgroup controller $\mathbf{S}_{(v-2),i}$ where $i = \lfloor \frac{j}{c} \rfloor$ and so on. It should be noted that $i = \lfloor \frac{k}{c^2} \rfloor$.

To illustrate the ContRi-Central Group Model, we present Figure 2.14 where there are 3 nodes in each subgroup (i.e. $c = 3$) and the height of the structure is $h = 3$. In Figure 2.14, there are 27 user nodes at the lowest layer $\mathcal{L}_2$ denoted as $\mathbf{S}_{2,i}$ for $0 \leq i \leq 26$. The subgroup controllers are in the higher layers $\mathcal{L}_1$ and $\mathcal{L}_0$. For example, the node $\mathbf{S}_{1,3}$ at $\mathcal{L}_1$ is the subgroup controller of subgroup $\mathcal{G}_{2,3} = \{\mathbf{S}_{2,9}, \mathbf{S}_{2,10}, \mathbf{S}_{2,11}\}$ at $\mathcal{L}_2$ while node $\mathbf{S}_{0,1}$ at $\mathcal{L}_0$ is the subgroup controller of subgroup $\mathcal{G}_{1,1} = \{\mathbf{S}_{1,3}, \mathbf{S}_{1,4}, \mathbf{S}_{1,5}\}$ at $\mathcal{L}_1$ respectively. It should be noted that the nodes $\mathbf{S}_{2,9}$ and $\mathbf{S}_{1,3}$, who are the first member of their respective subgroups $\mathcal{G}_{2,3}$ and $\mathcal{G}_{1,1}$, are two different nodes. The nodes

Figure 2.14: Nodes in the Contri-Central Group Model with $c = 3$ and $h = 3$.

$S_{1,3}$ and $S_{0,1}$ at layers $\mathcal{L}_1$ and $\mathcal{L}_0$ respectively are also different nodes.

Each subgroup $\mathcal{G}_{v,j}$ for $1 \leq v \leq h - 1$ and $0 \leq j \leq c^v - 1$ could also use different GKA protocols within their subgroup while the CG uses another GKA protocol to establish the final group key. In Chapter 8, we select the BD GKA protocol [BD95] to be used by the CG to establish the final group key K which is also the subgroup key $K_{0,0}$ of the CG while a variant of the NLKW protocol [NLKW05] will be used by each subgroup $\mathcal{G}_{v,j}$ for $1 \leq v \leq h - 1$ and $0 \leq j \leq c^v - 1$ to compute the subgroup key $K_{v,j}$. The BD protocol is chosen as it is very efficient, requiring only two rounds of communication and three modular exponentiations regardless of group size n . The variant of the NLKW protocol is chosen for the subgroups as it requires user nodes $S_{h-1,i}$ at the lowest layer $\mathcal{L}_{h-1}$ to compute only two modular exponentiations and one message transmission while each $S_{v,j}$ for $0 \leq v \leq h - 2$ and $0 \leq j \leq c^{v+1} - 1$, which do not have power constraints, computes $\mathcal{O}(c)$ exponentiations.

2.12.3 Centralized-Contributory Ring Group Model

Figure 2.15 shows the proposed Centralized-Contributory Ring (Central-Contri) group model that can be used to secure group communications for a heterogeneous network containing n users. Each subgroup in the Central-Contri group model can be either single layer such as subgroups $\mathcal{G}_1$ and $\mathcal{G}_4$ or multi-layer such as subgroups $\mathcal{G}_2$ and $\mathcal{G}_3$ in Figure 2.15. For the Central-Contri group model described in Chapter 9, we assume that there are only two layers and all subgroups $\mathcal{G}_i$ are single-

Figure 2.15: The Centralized-Contributory Ring (Central-ContRi) Group Model.

layered such as subgroups $\mathcal{G}_1$ and $\mathcal{G}_4$ in Figure 2.15. It should not be a problem to extend it to more layers. For the two-layer Central-ContRi group model, there are a total of n_c subgroups and in each subgroup $\mathcal{G}_i$, there are n_i members. The main controller $\mathcal{C}$ is located at the highest layer while the various subgroup controllers $\mathcal{S}_i$ together with the users nodes $\mathcal{U}_{i,j}$ for $2 \leq j \leq n_i$, which belongs to subgroup $\mathcal{G}_i$ for $1 \leq i \leq n_c$, are located at the second layer. It should be noted that $\mathcal{S}_i = \mathcal{U}_{i,1}$.

The difference between the Central-ContRi group model and the (C-H) group model described in Section 2.12.1 is that the Control Group in the Central-ContRi group model is using the star structure (i.e. there is an overall main controller $\mathcal{C}$) whereas the Control Group in the (C-H) group model is using a ring structure (i.e. no main controller $\mathcal{C}$).

Advantages: The advantages of the Central-ContRi group model are similar to the advantages of the (C-H) group model except that there is now a main controller $\mathcal{C}$ in the Control Group that can handle subgroup controller joining and leaving (at the Control Group Level) more efficiently.

Limitations: The limitations of the Central-ContRi group model are also similar to the limitations of the (C-H) group model except that the main controller could also be a single point of failure.

2.12.4 Centralized-Centralized Group Model

Figure 2.16 shows the Centralized-Centralized (Central-Central) group model. Each subgroup in the Central-Central group model can be either single layer such as subgroups $\mathcal{G}_1$ and $\mathcal{G}_4$ or multi-

Figure 2.16: Centralized-Centralized Group Model.

layer such as subgroups $\mathcal{G}_2$ and $\mathcal{G}_3$ in Figure 2.16. For the Central-Central group model described in Chapter 10, we assume that there are only two layers, i.e. all subgroups $\mathcal{G}_i$ are single-layered such as subgroups $\mathcal{G}_1$ and $\mathcal{G}_4$ in Figure 2.16. It should not be a problem to extend it to more layers. For the two-layer Central-Central group model, the main controller $\mathcal{C}$ is located at the highest layer while the various subgroup controllers $\mathcal{S}_i$ together with the users nodes $\mathcal{U}_{i,j}$, which belongs to subgroup $\mathcal{G}_i$ for $1 \leq i \leq n_c$, are located at the second layer.

The Central-Central group model can be easily implemented in WMNs and WiFi networks where the main controller $\mathcal{C}$ and subgroup controllers $\mathcal{S}_i$ are the wireless hot spots (WHSs) that do not have energy constraints while the user nodes $\mathcal{U}_{i,j}$ are mobile clients. The Central-Central group model can also be considered as a special case of the tree and star topology as mentioned in Section 2.9.2 and 2.9.4 respectively.

The difference between the Central-Central group model and the ContRi-Central group model described in Section 2.12.2 is that the Control Group in the Central-Central group model is using the star topology/structure (i.e. there is an overall main controller $\mathcal{C}$) whereas the Control Group in the ContRi-Central group model is using a ring topology/structure (i.e. no main controller $\mathcal{C}$).

Advantages: The advantages of the Central-Central group model are similar to the advantages of the ContRi-Central group model except that there is a main controller $\mathcal{C}$ in the Control Group of the Central-Central group model that can handle subgroup controller joining and leaving (at the Control Group Level) more efficiently.

Limitations: The limitations of the Central-Central group model are similar to the limitations of the ContRi-Central group model except that main controller $\mathcal{C}$ could be a single point of failure.

Chapter 3

Energy Analysis

3.1 Introduction

In this chapter, we discuss on the computational and communication energy analysis used to analyze the energy costs of running GKM protocols for typical user nodes in wireless networks. User nodes in wireless networks are usually low power energy constrained devices such as PDAs that run on batteries or in the case of wireless sensor networks, low power energy constrained sensor nodes with limited battery resources that usually cannot be recharged or replaced. Therefore, there is a need to design GKM protocols that are more energy-efficient. However, most current GKM protocols only provide the complexity analysis of GKM protocols and do not discuss the energy consumption costs. To provide a better understanding of the energy consumption costs of user devices in wireless networks, we also provide energy analysis for the GKM protocols developed in this thesis. We shall examine the computational energy costs of two variants of the SA-1110 “StrongARM” microprocessor, namely the 133MHz and 206MHz “StrongARM” microprocessors. We chose the 133MHz StrongARM microprocessor for our computation analysis because it is a widely used and popular microprocessor that can be found in a wide range of devices such as wireless sensor nodes (e.g. Rockwell Scientific WINS nodes), single-board computers (SBCs) (e.g. Forth-Systeme DIMM-520, ModNET50 SBC, Intrinsyc CerfBoard SBC) and also certain models of handheld computers and PDAs (e.g. Psion Series 7 handheld computer and Invair Filewalker PDA

that is the size of a credit card). Furthermore, the energy analysis of the 133MHz StrongARM microprocessor has also been in active research and discussed in several papers [CKM00,SC01,HV02,GST07]. We also examine a faster version of the StrongARM microprocessor, namely the 206MHz “StrongARM” microprocessor. The 206MHz “StrongARM” microprocessor was chosen because it is a very popular microprocessor for PDAs and handheld computers, which form the majority of the low-power energy constrained user nodes in wireless networks. For example, the 206MHz “StrongARM” microprocessor is used in many PDAs such as Compaq iPAQ H3670 PDA, PPT2800 Pocket PC device, Sharp Zaurus SL-5x00, CDL Paron ‘secure PDA’, CIIT multimedia PDA, G.Mate Yopy, HP Jornada 720, GSPDA V-2002, HanGil C3224 multimedia PDA, HNT Exilien handheld multimedia PCs, MasterIA PDA, Mizi Linux PDA developer kit, Royal Lin@x and SK Telecom IMT2000 WebPhone. Besides, the energy analysis of the 206MHz “StrongARM” microprocessor has also been widely researched and published in several papers [KM03,PRRJ03,PRRJ06]. For the communication energy analysis, we shall discuss on three different types of wireless communication devices, namely the 100kbps radio transceiver module used on the Rockwell Scientific WINS sensor nodes [ACPR99], the IEEE 802.11 Spectrum24 LA-4121 WLAN card commonly used by PDAs for wireless communications and Lucent IEEE 802.11 11Mbps WaveLAN card that is also used by PDAs and laptop computers for communications in wireless networks. The 100kbps radio transceiver module was selected for communication analysis because it represents a low power radio transceiver that is used in small wireless network devices such as sensor nodes while the IEEE 802.11 Spectrum24 LA-4121 WLAN card and Lucent IEEE 802.11 11Mbps WaveLAN card are chosen as they represent two popular WLAN cards that are used by user devices in wireless networks such as PDAs, handheld computers and laptop computers for wireless communications. Besides, the energy analysis of both WLAN cards have been studied and published in [KM03] and [FN01] respectively.

3.2 Current Work in Literature

The detailed report by Carman et al [CKM00] in 2000 provided energy consumption analysis of some GKA protocols in distributed sensor networks. Besides this, the detailed report [CKM00] also provided computational energy costs for public key authentication algorithms, symmetric key

cryptography algorithms and hashing algorithms for different low-power embedded microprocessors. It also provides the energy consumption costs for the communication module used in wireless sensor nodes such as the Wireless Integrated Network Sensor (WINS) node by Sensoria. However, the report only considered the total energy consumption costs for communications and did not discuss the transmission and reception of messages separately. The transmission energy cost per bit is usually twice of the reception energy cost per bit, but certain GKA protocols may require users to transmit fewer messages and receive large amounts of messages. Hence the communication energy costs should be discussed separately in order to determine which communication costs (Transmission or Reception) consumes more energy and should be reduced.

Heinzelman et al. provided models for predicting quality and energy for wireless microsensor networks and showed the advantages of trading off these two parameters in their paper [HSWC00]. In 2002, Hodjat and Verbauwhede [HV02] presented the energy cost results for AES [dae02] secret-key algorithm and Elliptic Curve Cryptography (ECC) [IEE99] in ad hoc networks. They also showed the energy consumption costs of key-exchange using Diffie Hellman (DH) [DH76] based on ECC and Kerberos [SNS88, KN93] based on AES. Their work was based on the WINS node developed by Rockwell Scientific. In [KM03], the authors measured the energy consumed by secure wireless sessions on a mobile test bed that consists of a Pocket PC device equipped with an 11 Mbps Spectrum24 wireless LAN (WLAN) adapter card. They also provided the power consumption values of the 11 Mbps Spectrum24 WLAN card in their paper. Wander et al also studied the energy analysis of public-key cryptography (RSA and ECDSA) in wireless sensor networks. Their work is based on an 8-bit microcontroller platform.

In 2001, Feeney and Nilsson [FN01] did a comprehensive study on the Lucent 11MBps WLAN card and provided the P2P and broadcast communication (transmission and reception) energy costs. Most recently in 2006, Potlapally et al [PRRJ06] presented the energy costs of the Secure Socket Layer (SSL) protocol using an iPAQ PDA equipped with a WLAN card. However, they only provided the total energy consumption of the iPAQ PDA and not the detailed energy costs of the 206 MHz “StrongARM” microprocessor that is performing the computations on the iPAQ PDA. Furthermore, their work is based on the SSL protocol and not GKA protocols and they did not analyze P2P and broadcast communications separately. Also in 2006, Groszschädl et al [GST07]

evaluated the energy cost of running two key exchange protocols, namely Kerberos [SNS88,KN93] and ECMQV [LMQ⁺03], on the Rockwell WINS sensor node. Their evaluation considers both the processor's energy consumption for calculating cryptographic primitives and also the energy costs of radio communications of the WINS node. Piotrowski et al [PLP06] investigated the energy consumption for Public Key Cryptography (common RSA and ECC operations) for four types of sensor nodes. Both [GST07] and [PLP06], together with [CKM00], focus only wireless sensor networks, where all nodes are small low-power devices.

3.3 Computational Energy Cost Analysis

In this section, we present the energy costs of the 133MHz and 206MHz SA-1110 "StrongARM" Microprocessor. For the 133MHz SA-1110 "StrongARM" Microprocessor, our results are based on [CKM00] and we show a general formula that can be used to obtain the energy costs of different cryptographic operations such as DSA signature generation and verification operations. Next, using the computational timing costs obtained from the MIRACL software library [Mir] and extrapolation, we estimate the energy costs of various different cryptographic operations such as Elliptic curve Tate Pairing and Scalar Multiplication on the 133MHz SA-1110 "StrongARM" Microprocessor. We show our results in Table 3.2 for the 133MHz SA-1110 "StrongARM" Microprocessor. Next, we study on the 206MHz SA-1110 "StrongARM" Microprocessor which shares the same architecture as the 133MHz "StrongARM" Microprocessor discussed earlier [Int]. Using the information from the datasheet for the SA-1110 "StrongARM" Microprocessor [Int], we compute the energy costs for various cryptographic operations when using the 206MHz SA-1110 "StrongARM" Microprocessor and present our results in Table 3.3.

3.3.1 133MHz SA-1110 "StrongARM" Microprocessor

In this section, we present the computational energy costs for the 133MHz SA-1110 "StrongARM" microprocessor that is typically used in user devices such as PDAs in wireless networks and also used in the Rockwell Scientific WINS sensor nodes in wireless sensor networks. The energy consumption values of the 133MHz SA-1110 "StrongARM" microprocessor are based on studies conducted by [CKM00]. The low-power StrongARM microprocessor runs at 133MHz, consumes less than

300mW in the run mode and less than 1mW in the sleep mode. As mentioned in [CKM00], the StrongARM microprocessor consumes 9.1mJ for a 1024-bits Diffie-Hellman (DH) [DH76] operation with 160-bits exponent, 9.1mJ for a DSA signature generation and 18.2mJ for DSA signature verification (DSA certificate verification also consumes 18.2mJ). In general, the energy consumed for performing a cryptographic operation X (e.g. RSA encryption) $E_{X(133MHZ)}$ for processor Y can be computed using the general formula [CKM00]:

$$E_{X(133MHZ)} = N_{128-bit_X} \cdot \frac{CLK_{128-bit_X}}{freq_Y} \cdot P_Y \quad (3.1)$$

where $N_{128-bit_X}$ refers to the number of 128-bits multiplication operations required for cryptographic operation X , $CLK_{128-bit_X}$ refers to the number of clocks required for a 128-bits multiplication operation, $freq_Y$ refers to the clock frequency of Processor Y (e.g. 133MHz for the StrongARM processor) and P_Y refers to the power consumption of Processor Y (e.g. 240mW for the StrongARM processor). For brevity, we refer readers to [CKM00] for the formula to compute $N_{128-bit_X}$ as well as the clock frequency and power consumption of several other processors that are commonly used in mobile devices. By using equation (3.1), the energy consumed for performing cryptographic operation X on different microprocessors Y can be obtained and analyzed.

Energy consumption costs for other cryptographic computations

As some cryptographic operations such as Map-To-Point, Tate Pairing and Scalar Multiplication, ECDSA and SOK signature generation and verification are not given in [CKM00], we estimate the energy costs of these operations using the extrapolation method by using results from [CKM00] for the 133MHz StrongARM microprocessor and the MIRACL software library [Mir] for the Pentium III 450MHz (P3-450MHz) microprocessor that contains the timing costs of these cryptographic operations. We shall now discuss on the extrapolation method as follows:

As the power consumption for the 133MHz StrongARM microprocessor is 240mW [CKM00, Int], we have the general formula to compute $\alpha_{X(133MHz)}$, the timing costs in milliseconds (ms) for performing cryptographic operation X on the 133MHz StrongARM microprocessor:

$$\alpha_{X(133MHz)} = \frac{E_{X(133MHZ)} (mJ)}{240mW} \quad (3.2)$$

Table 3.1: Costs of Cryptographic Operations on 133MHz StrongARM microprocessor and P3-450MHz microprocessor

Cryptographic Operations	133MHz StrongARM		Pentium III 450MHz
	Energy Costs	Timing Costs	Timing Costs
Modular Exponentiation	9.1mJ	37.92ms	8.8ms
RSA signature generation	15mJ	62.5ms	17ms
RSA signature verification	0.74mJ	3.08ms	0.81ms
DSA signature generation	9.1mJ	37.92ms	8.8ms
DSA signature verification	18.2mJ	75.83ms	17.6ms

Table 3.1 shows the energy costs and timing costs of various cryptographic operations on the 133MHz StrongARM microprocessor and the timing costs of these cryptographic operations on the P3-450MHz microprocessor. The energy costs in the first column for the various cryptographic operations on the 133MHz StrongARM microprocessor is from [CKM00]. Using Equation (3.2), we obtain the timing costs for the various cryptographic operations on the 133MHz StrongARM microprocessor as shown in the second column. The timing costs shown in the last column are from the MIRACL software library [Mir] for the various cryptographic operations on the P3-450MHz microprocessor. Using the second and third column, we can plot an extrapolation graph as shown in Figure 3.1 where the Y-axis and X-axis shows the timing costs of various cryptographic operations on the 133MHz StrongARM microprocessor and P3-450MHz microprocessor respectively. As shown in Figure 3.1, the R-Squared value (R^2) is 0.9858, which means that the trendline is a very good fit to the data. Furthermore, $R^2 = 0.9858$ also means that our extrapolation results are good predictions because if the R^2 value is 1.0, then you can perfectly predict the value of another term given the value of one term.

From Figure 3.1, we can obtain the extrapolation equation to compute $\alpha_{X(133MHz)}$, the timing costs in milliseconds (ms) for performing cryptographic operation X on the 133MHz StrongARM microprocessor, to be:

$$\alpha_{X(133MHz)} = 4.0385 \cdot \gamma_{X(P3-450)} \quad (3.3)$$

where $\gamma_{X(P3-450)}$ refers to the time taken for cryptographic operation X on the P3-450MHz microprocessor. Rearranging Equation (3.2), we can also obtain the energy costs of cryptographic

Figure 3.1: Extrapolation

operation X on the 133MHz StrongARM microprocessor to be:

$$E_{X(133MHz)} = 240mW \cdot \alpha_{X(133MHz)} \quad (3.4)$$

Using Equations (3.2), (3.3), (3.4) and the timing costs from the MIRACL software library [Mir] for cryptographic operations such as Map-To-Point, Tate Pairing and Scalar Multiplication, ECDSA and SOK signature generation and verification that are not discussed in [CKM00], we are able to estimate the energy costs of these cryptographic operations as shown in Table 3.2. The timing cost of the Tate Pairing operation was given to be 20ms on the Pentium III 1GHz (P3-1GHz) microprocessor [Mir]. To obtain the equivalent timing cost on the P3-450MHz microprocessor, we scale down by a factor of $\frac{1000MHz}{450MHz} = 2.22$ to obtain 44.4ms. From the MIRACL software library [Mir], the timing costs of Identity-based Encryption (IBE) Encryption and Decryption were given to be 35ms and 27ms respectively on the P3-1GHz. As the IBE Encryption requires one additional Map-To-Point operation than IBE Decryption, we were able to estimate the timing cost of Map-To-Point operation to be $35 - 27 = 8ms$. To get the timing cost of the Map-To-Point operation on the P3-450MHz, we scale down by a factor of 2.22 to get 17.78ms.

The results in Table 3.2 are used in the energy analysis of some of the GKM protocols proposed

Table 3.2: Computational Energy Cost for 133MHz SA-1110 “StrongARM” Microprocessor

		133MHz StrongARM		450MHz P-III
Modular Exponentiation		9.1mJ	37.92ms	8.8ms
Map-To-Point**		17.2mJ	71.80ms	17.78ms
Tate Pairing**		43.0mJ	179.3ms	44.4ms
Scalar Multiplication**		8.2mJ	34.33ms	8.5ms
Signature Generation	RSA	15mJ	62.5ms	17ms
	DSA	9.1mJ	37.92ms	8.8ms
	ECDSA**	8.2mJ	34.33ms	8.5ms
	SOK**	16.5mJ	68.66ms	17ms
	GQ	18.2mJ	75.83ms	17.6ms
Signature Verification	RSA	0.74mJ	3.08ms	0.81ms
	DSA	18.2mJ	75.83ms	17.6ms
	ECDSA**	16.5mJ	68.66ms	17ms
	SOK**	129.1mJ	537.93ms	133.2ms
	GQ	18.2mJ	75.83ms	17.6ms

** denotes results obtained through extrapolation.

For RSA signature scheme, $e = 65537$.

in this thesis.

3.3.2 206 MHz SA-1110 “StrongARM” Microprocessor

In the previous section, the energy costs of various cryptographic operations on the 133MHz “StrongARM” microprocessor were given. However, some current PDAs are powered by faster microprocessors (e.g. the Compaq iPAQ H3670 PDA is powered by the 206 MHz “StrongARM” microprocessor). Although Potlapally et al [PRRJ06] presented the energy analysis of some cryptographic operations on the iPAQ PDA, but they measured the energy costs of the iPAQ PDA as a whole, which includes energy consumed by other modules such as the LCD display driver etc. To obtain the time taken and energy costs of the various cryptographic operations on the 206 MHz “StrongARM” microprocessor as shown in Table 3.3, we use the energy model based on Carman et al’s report [CKM00] by first calculating the clock cycles of the four operations shown in Table 3.3 using the information (energy costs for the four cryptographic operations, power consumption and speed of the 133MHz “StrongARM” microprocessor) from Carman et al’s report [CKM00] and the

Table 3.3: Computational Energy Costs for 206MHz SA-1110 “StrongARM” Microprocessor

	Time taken	Energy Consumed
Modular Exponentiation	24.48ms	9.792mJ
RSA Sign. Generation	40.35ms	16.14mJ
RSA Sign. Verification	1.99ms	0.796mJ
RSA Cert. Verification	1.99ms	0.796mJ

formula:

$$E_X = P_Y \times \frac{CLK_X}{freq_Y} \quad (3.5)$$

where E_X is energy cost and CLK_X is the number of clock cycles to perform cryptographic operation X on Processor Y respectively while P_Y and $freq_Y$ refers to the power consumption and speed of Processor Y respectively. Since both the 133MHz and 206 MHz are “StrongARM” microprocessors [Int], the time taken for the four cryptographic operations in Table 3.3 on the 206 MHz “StrongARM” microprocessor can be obtained using the formula:

$$time_X = \frac{CLK_X}{freq_Y}$$

where $time_X$ is the time taken for cryptographic operation X on Processor Y and the speed $freq_Y$ is 206 MHz for the 206 MHz “StrongARM” microprocessor. As the power consumption for the 206 MHz “StrongARM” microprocessor is given to be 400mW from [Int], we can obtain the energy costs shown in Table 3.3 using Equation (3.5) and the clock cycles for the four cryptographic operations computed earlier.

3.4 Communication Energy Cost Analysis

In this section, we shall examine the communication energy costs of three different types of wireless communication devices, namely the 100kbps radio transceiver module used on the Rockwell Scientific WINS sensor nodes, the IEEE 802.11 Spectrum24 LA-4121 WLAN card commonly used by PDAs for wireless communications and Lucent IEEE 802.11 11Mbps WaveLAN card that is also used by PDAs and laptop computes for communications in wireless networks. We first examine the 100kbps transceiver module and IEEE 802.11 Spectrum24 LA-4121 WLAN card and based

our studies on the using the information given in [CKM00] and [HV02] for the 100kbps transceiver module and [KM03] for the WLAN card. We also provided some transmission and reception energy costs formulas which can be used to compute the communication energy costs of different communication radio transceiver modules. Next, we discuss on the Lucent IEEE 802.11 11Mbps WaveLAN card using studies conducted by Feeney and Nilsson in [FN01]. We present the energy model that can be used to compute more accurate transmission and reception energy costs of messages of different size sent and received using the Lucent IEEE 802.11 11Mbps WaveLAN card respectively.

3.4.1 100kbps transceiver module and IEEE 802.11 Spectrum24 LA-4121 WLAN card

In this section, we show the communication energy costs of the 100kbps radio transceiver module used on the Rockwell Scientific WINS sensor nodes and the IEEE 802.11 Spectrum24 LA-4121 WLAN card commonly used by PDAs for wireless communications. For the 100kbps radio transceiver module, the energy costs are based on the information given in [CKM00] and [HV02] and calculated based on maximum transmission power level and at a transmission rate of 100kbps. The transmission range for the 100kbps radio transceiver module is approximately 100 meters [CKM00, CSW03]. The energy consumption for transmission is $10.8\mu J/bit$ and the energy consumption for reception is $7.51\mu J/bit$ for the 100kbps radio transceiver module. For the IEEE 802.11 Spectrum24 LA-4121 WLAN card, we based our energy costs from the information given by [KM03] and the data transmission and reception rates are assumed to be identical at 200Kbps. The transmission range for the Spectrum24 LA-4121 WLAN card is 90 meters for indoor range and 450 meters for outdoor range from the Spectrum24 LA-4121 WLAN card datasheet [spe]. The energy consumption for transmission is $0.658\mu J/bit$ and the energy consumption for reception is $0.305\mu J/bit$ for the IEEE 802.11 Spectrum24 LA-4121 WLAN card. The energy consumption for the IEEE 802.11 Spectrum24 LA-4121 WLAN card is based on the transmission power level of 2.05W and reception power level of 0.95W respectively as mentioned in [KM03]. Next, we describe the generalized transmission and reception energy cost formulas that can be used to obtain the energy costs of various cryptographic operations.

Transmission Energy Cost Formula

In general, the transmission energy cost $E_{Tx}(B)$ for B -bits can be computed using the formula:

$$E_{Tx}(B) = B \cdot \varepsilon_{Tx} \quad (3.6)$$

where ε_{Tx} refers to the energy consumed by the communication radio to transmit one bit (J/bit). A more detailed model for transmission is given in [HSWC00]. In that model, to transmit a B -bits message over a distance d , the radio expends:

$$E_{Tx}(B, d) = B \cdot \varepsilon_{Tx} + B \cdot d^2 \cdot \varepsilon_{Tx-amp} \quad (3.7)$$

where ε_{Tx-amp} is the energy consumed per bit per square meter ($J/bit/m^2$) for the transmit amplifier to achieve an acceptable signal to noise ratio (E_b/N_0).

Reception Energy Cost Formula

To receive a B -bits message, the reception energy cost $E_{Rx}(B)$ could be computed using the formula:

$$E_{Rx}(B) = B \cdot \varepsilon_{Rx} \quad (3.8)$$

where ε_{Rx} refers to the energy consumed by the radio to receive one bit (J/bit).

By using equation (3.6) or (3.7) for transmission and equation (3.8) for reception, we can also analyze the energy consumed by different communication radio transceiver modules.

By using the transmission and reception energy per bit values for the 100kbps transceiver module and IEEE 802.11 Spectrum24 LA-4121 WLAN card respectively, together with equation (3.6) or (3.7) for transmission and equation (3.8) for reception, we obtain Table 3.4, which shows the transmission and reception energy costs of certificates and signatures for different signature schemes.

Table 3.4: Communication Energy Cost for 100kbps transceiver module and IEEE 802.11 Spectrum24 LA-4121 WLAN card

	100kbps Transceiver	WLAN Card
Transmission per bit	$10.8\mu J$	$0.66\mu J$
Reception per bit	$7.51\mu J$	$0.31\mu J$
Transmission of 172-Bytes DSA certificate	$14.86mJ$	$0.905mJ$
Reception of 172-Bytes DSA certificate	$10.33mJ$	$0.42mJ$
Transmission of 86-Bytes ECDSA certificate	$7.43mJ$	$0.45mJ$
Reception of 86-Bytes ECDSA certificate	$5.17mJ$	$0.21mJ$
Transmission of DSA/ECDSA signature ¹	$3.46mJ$	$0.21mJ$
Reception of DSA/ECDSA signature ¹	$2.40mJ$	$0.1mJ$
Transmission of SOK signature ²	$4.19mJ$	$0.26mJ$
Reception of SOK signature ²	$2.91mJ$	$0.12mJ$
Transmission of GQ signature ³	$12.79mJ$	$0.78mJ$
Reception of GQ signature ³	$8.89mJ$	$0.36mJ$

¹ DSA/ECDSA signature (r, s) , both $r, s = 160$ -bits.

² SOK signature (S_1, S_2) , both $S_1, S_2 = 194$ -bits.

³ GQ signature (s, c) , $s = 1024$ -bits and $c = 160$ -bits.

3.4.2 Lucent IEEE 802.11 11Mbps WaveLAN card

In Table 3.5, we show the communication energy costs formulas for point-to-point (P2P) and broadcast transmission and reception for the Lucent IEEE 802.11 11Mbps WaveLAN card that is taken from [FN01]. Unlike most current communication energy analysis that do not differentiate between P2P and broadcast communications, our communication energy analysis separates energy calculations for P2P and broadcast communications to give better understanding of the actual communication energy costs in GKA. The communication energy costs in Table 3.5 are represented using a linear equation:

$$Energy = m \times size + b$$

where m is the incremental component which is proportional to the message size $size$ and b is the fixed component associated with device state changes and channel acquisition overhead. For example, $m = 2.1$ and $b = 272$ for broadcast send as shown in Table 3.5. The communication energy costs in Table 3.5 are measured in microJoules (μJ) and message size $size$ are measured in bytes. Experimental results in [FN01] confirmed the accuracy of the linear equation model and were used to determine the values for the linear coefficients m and b for P2P and broadcast send

(transmission) and receive (reception) respectively as shown in Table 3.5.

Table 3.5: Communications Energy Costs for Lucent IEEE 802.11 11Mbps WaveLAN card

	$\mu W \cdot sec/byte \times byte + \mu W \cdot sec$
Point-to-point (P2P) send	$0.48 \times size + 431$
Broadcast send	$2.1 \times size + 272$
Point-to-point (P2P) receive	$0.12 \times size + 316$
Broadcast receive	$0.26 \times size + 50$

From the Lucent WaveLAN card datasheet [Luc], the transmission range for the Lucent WaveLAN card is between 200 to 250 meters.

3.4.3 Certificate Size

In this section, we describe the size and fields for the RSA, DSA and ECDSA certificate respectively used in our analysis. Table 3.2 shows the fields in a X.509 (Version 3) certificates. To improve efficiency and reduce the size of the certificate, we simplified the X.509 certificate for use in our analysis. For the RSA, DSA and ECDSA certificates, we only keep three important fields which are **Public Key Information**, **Subject Unique ID** and the signature from a Certification Authority (CA) of the issued certificate (under the **Extensions** field). In our GKA protocol analysis, we assume that the Subject Unique ID is sufficient for identification of the node and the certificate issued to the node. We remove the following fields: Issuer Name, Issuer Unique ID and Validity Period as we assume that all certificates have a fixed validity period and that a single authority is responsible for managing the network. The Signature Algorithm Identifier field is removed since only one specific cipher suite (e.g. AES) and one signature algorithm (e.g. RSA or DSA or ECDSA) are used in the GKA protocol. Finally, the Serial Number and Version fields are removed as we assume that all network nodes implement the same certificate version.

In the following subsections, we describe the simplified X.509 certificates used for RSA, DSA and ECDSA signature schemes respectively in our analysis.

Version
Serial Number
Signature Algorithm Identifier
Issuer Name
Validity Period
Subject Name
Public Key Information
Issuer Unique ID
Subject Unique ID
Extensions

Figure 3.2: X.509 (Version 3) certificate

RSA Certificate

The 263-bytes (2104-bits) RSA certificate cert_x , where $x = C, i, ij$, used in the proposed GKA protocols and in the complexity and energy analysis comprises of a 4-bytes (32-bits) unique Identity (ID) to identify the node to which the certificate is issued to, a 128-bytes (1024-bits) RSA modulus $\bar{N}_x$, a 3-bytes (24-bits) RSA public exponent e and a 128-bytes (1024-bits) RSA signature of certificate cert_x from a Certification Authority (CA). This 263-bytes RSA certificate is a simplified version of the X.509-compliant (V.3) certificate.

DSA Certificate

The 172-bytes DSA certificate comprises of a 4 byte (32 bits) unique Identity (ID) to identify the node to which the certificate is issued to, a 128 bytes (1024 bits) public key and a 40 bytes (320 bits) DSA signature from a Certificate Authority (CA). This 172-bytes DSA certificate is a simplified version of the X.509-compliant (V.3) certificate.

ECDSA Certificate

The 86-bytes ECDSA certificate comprises of a 4 byte (32 bits) unique Identity (ID) to identify the node to which the certificate is issued to, a 40 bytes ((320 bits)) ECDSA public key and a 42 bytes (336 bits) ECDSA signature from a Certificate Authority (CA). This 86-bytes ECDSA certificate is a simplified version of the X.509-compliant (V.3) certificate.

3.5 Summary

In this chapter, we examine the computational energy costs of two variants of the SA-1110 “StrongARM” microprocessor, namely the 133MHz and 206MHz “StrongARM” microprocessors. Next, we looked at communication energy costs and discussed three different wireless communication devices, namely the 100kbps radio transceiver module used on sensor nodes, the IEEE 802.11 Spectrum24 LA-4121 WLAN card and Lucent IEEE 802.11 11Mbps WaveLAN card that are used by PDAs and laptop computers for wireless communications in wireless networks.

Chapter 4

Authenticated Group Key Agreement for Wireless Networks

4.1 Introduction

Authentication is a very important feature as messages in wireless networks are sent over the air. Therefore, any malicious adversary can simply pretend to be a legitimate user node and join the group key agreement (GKA) to obtain the shared group key.

The most common way of providing authentication in GKA protocols is through the use of digital signatures. However, this approach usually requires each group member to verify all the messages received. Therefore, if a member received n messages where n is the group size, n signature verifications would have to be performed, which would result in significant computational cost if n is large. In some cases, an extra round of communication may have to be introduced.

As broadcast is the natural mode of communication in wireless networks, the efficient and secure Burmester-Desmedt (BD) GKA protocol [BD95] described in Section 2.7.6 is suitable for use in wireless networks as the BD protocol requires only *two* rounds of communication and only *one* message to be broadcast by each member in each round, regardless of the number of group members. Furthermore, in wireless networks such as mobile ad hoc networks (MANETs) and wireless sensor networks (WSNs) where all user nodes have equal computational powers, the BD protocol is even more suitable because the BD protocol is fair to all nodes since every node only has to compute two modular exponentiations and send two messages each.

In that same paper [BD95], Burmester and Desmedt also proposed an authenticated GKA protocol. However, Just and Vaudenay [JV96] showed that the proposed authenticated protocol does not provide group key authentication. The current intuitive way of providing authentication for the BD protocol is to sign and verify all messages sent and received respectively. Another method of providing authentication for the BD is introduced by Katz and Yung [KY03]. However, this would result in an additional round of communication. Both approaches suffer the same problem of large computational cost of signature verification if the number of group members is big.

In this chapter, we first present a unique characteristic of the BD protocol after careful study of the BD protocol. Next, we describe the current techniques used to provide authentication for the BD protocol. We then present an efficient authenticated GKA protocol that is based on a variant of the Schnorr signature scheme [Sch91] and the BD protocol [BD95]. We show that the proposed authenticated protocol does not increase the number of communication rounds. The proposed protocol is also more efficient as it requires each member to generate only *one* signature and perform only *one* signature verification regardless of the group size. A security analysis shows that the proposed protocol is secure against malicious adversaries and is also secure against insider impersonations where k ($< n - 1$) malicious group members (i.e. k malicious insider nodes) collude with each other to impersonate as a particular honest node, say U_i . The complexity analysis of two methods of authentication for the BD and the proposed protocol and show that the proposed protocol is more efficient in comparison.

4.2 Characteristic of Burmester-Desmedt protocol (BD)

In this section, we first present an unique characteristic of the BD protocol. In the BD protocol as mentioned in Section 2.7.6, we have $X_i = g^{r_i r_{i+1} - r_{i-1} r_i} \bmod p$ for $1 \leq i \leq n$ where $r_0 = r_n$ and $r_{n+1} = r_1$. Therefore, we have the following lemmas:

Lemma 2 $\prod_{i=1}^n X_i = 1 \bmod p$.

Lemma 3 For each member U_k in the group for $1 \leq k \leq n$, the member U_k can obtain $g^{r_j - 1 r_j} \bmod p$ for $1 \leq j \leq n$, where $r_0 = r_n$.

Proof: As given $X_i = g^{r_i r_{i+1} - r_{i-1} r_i} \bmod p$ for $1 \leq i \leq n$ where $r_0 = r_n$ and $r_{n+1} = r_1$ and for any $j < n$, we consider the following two cases:

Case (i) $j < k$: Then, compute $X = \prod_{i=j}^k X_i \bmod p = g^{r_k r_{k+1} - r_{j-1} r_j} \bmod p$. Since, the member U_k knows r_k , then he also knows $g^{r_k r_{k+1}}$. Therefore, the member U_k obtains $g^{r_{j-1} r_j} = X^{-1} \cdot g^{r_k r_{k+1}} \bmod p$ (if $k = n$, then $r_{k+1} = r_1$).

Case (ii) $j > k$: Then, compute $X = \prod_{i=k}^j X_i \bmod p = g^{r_j r_{j+1} - r_{k-1} r_k} \bmod p$ (if $j = n$, then $r_{j+1} = r_1$). Since, the member U_k knows r_k , then he also knows $g^{r_{k-1} r_k}$. Therefore, the member U_k obtains $g^{r_j r_{j+1}} = X \cdot g^{r_{k-1} r_k} \bmod p$ (if $j = n$, then $r_{j+1} = r_1$).

Hence, the member U_k could obtain $g^{r_{j-1} r_j}$ for all $1 \leq j \leq n$ where $r_0 = r_n$.

From Lemma 3, we know that each member can obtain $g^{r_{j-1} r_j}$ for all $1 \leq j \leq n$ for any given X_i for $1 \leq i \leq n$ where $r_0 = r_n$. Therefore, each member can compute the agreed group key in BD as $K = \prod_{j=1}^n g^{r_{j-1} r_j}$ where $r_0 = r_n$. Hence, it is not necessary to carry out another exponentiation as in equation (2.3) in Chapter 2 Section 2.7.6. This will reduce by *one* the number of exponentiations required to compute group key K in BD .

4.3 Authentication Techniques for Burmester-Desmedt (BD)

In this section, we describe two current methods of providing authentication for the BD. The first method is the intuitive way of signing and verifying all messages sent and received respectively. Another method is to add an additional round of communication. Both approaches have the same problem of large computational costs of signature verification when the number of group members is large. We describe the two methods below.

4.3.1 Authenticated BD using digital signatures

The original BD [BD95] described in Section 2.7.6 does not provide authentication. One common way of providing authentication for the BD is by the intuitive method also mentioned in papers [AKNR⁺04] and [KPT04b] of signing and verifying all the messages sent and received respectively. This method is described as follows:

During Round 1, each group member U_i for $1 \leq i \leq n$ signs z_i and broadcast z_i and its signature. During Round 2, each U_i first verifies the signatures on z_{i+1} and z_{i-1} as they are necessary for the

computation of X_i . If the two signature verifications are correct, then U_i proceeds to compute X_i . Each U_i then signs X_i and broadcast X_i and its signature in Round 2. Before key computation, each U_i verifies all the received X_j for $1 \leq j \neq i \leq n$, resulting in the verification of $n - 1$ signatures, where n is the group size. Therefore, each group member U_i has to generate *two* signature and verify $n + 1$ signatures using this approach.

4.3.2 Authenticated BD using an extra round

Another method of providing authentication for the BD is to apply an extra round of communication. This method has also been used in papers [KY03] and [NLKW04]. The advantage of this method is that it provides a modular approach that simplifies the design and analysis of authenticated GKA protocols. The extra round also transforms the BD protocol secure against a passive adversary to one that is secure against a stronger active adversary that controls all communications in the network. The extra round of communication is known as the initialization phase that is to be executed before the protocol is ran for the first time. This method is described as follows:

During the initialization phase, each group member U_i begins by choosing an instance identifier (IID) $\phi_i \in \{0, 1\}^l$ where l is an integer and broadcast $\{U_i, \phi_i\}$. After receiving all the $n - 1$ IIDs from other group members, each U_i stores $\Phi_i = \{U_i, \phi_i\}$ for $1 \leq i \leq n$.

In the GKA phase, instead of sending m_i where

$$m_i = \begin{cases} z_i & \text{in Round 1 and,} \\ X_i & \text{in Round 2 of original BD protocol} \end{cases}$$

U_i sends $m'_i = (m_i, \sigma_i)$, where σ_i is the signature of $m_i || \Phi_i$, and $||$ denotes concatenation.

Upon receiving a message $m'_j = (m_j, \sigma_j)$ from member U_j , U_i verifies that m'_j is correct by performing a signature verification on $m_j || \Phi_i$ and comparing this with σ_j . Note that this method of signature verification is different from the method described above (authenticate BD using digital signatures), where a signature verification is performed on m_j instead of $m_j || \Phi_i$.

Hence, for authenticated BD using an extra round, Round 1 refers to the initialization phase as mentioned above. In Round 2, each U_i sends (z_i, σ_i) where σ_i is the signature of z_i . In Round 3, U_i first verifies the two messages received from his neighbors containing z_{i+1} and z_{i-1} before computing

X_i . For example, U_i receives (z_{i+1}, σ_{i+1}) from U_{i+1} . U_i then performs a signature verification on $z_{i+1} || \Phi_i$ and checks this with σ_{i+1} . U_i accepts z_{i+1} if the check is correct.

After computing X_i , each U_i broadcasts (X_i, σ'_i) where σ'_i in this case is the signature on X_i . Similarly, before computing the final group key, U_i has to verify all $n - 1$ messages containing X_j for $1 \leq j \neq i \leq n$. This results in two signature generation and $n + 1$ signature verifications.

4.4 Proposed Authenticated Group Key Agreement Protocol

In this section, we present the proposed authenticated GKA protocol that combines the computational-efficiency of the Schnorr signature scheme [Sch91] and the round-efficiency of the BD [BD95] to provide an efficient, authenticated and secure GKA protocol suitable for secure group communication in wireless networks. The proposed protocol requires each group member U_i to generate only *one* signature and perform only *one* signature verification. Like the original BD, the proposed protocol requires only *two* rounds of communication. Let p (1024 bits) and q (160 bits) be prime numbers such that q divides $p - 1$ and H be a one-way hash function. Next, an element $g \in Z_p^*$ of order q is selected. We describe the proposed protocol in detail.

Each group member U_i for $1 \leq i \leq n$ has two parameters namely x_i , the private key of U_i , where $x_i \in Z_q^*$ and y_i , the public key of U_i , where $y_i = g^{x_i} \bmod p$.

Round 1: Each U_i for $1 \leq i \leq n$, selects $r_i, k_i \in Z_q^*$ and computes $z_i = g^{r_i} \bmod p$ and $\tau_i = g^{k_i} \bmod p$. Finally, U_i broadcast z_i, τ_i and its public key y_i to all other group members.

Round 2: After receiving the information from Round 1, each U_i for $1 \leq i \leq n$ will compute $c_i = H(z_i)$, $c_{i+1} = H(z_{i+1})$, $c_{i-1} = H(z_{i-1})$ and

$$\hat{X}_i = \left(\frac{z_{i+1} \cdot y_{i+1}^{c_{i+1}}}{z_{i-1} \cdot y_{i-1}^{c_{i-1}}} \right)^{r_i + c_i x_i} \bmod p \quad (4.1)$$

Next, U_i computes $\mathcal{Z} = \prod_{j=1}^n z_j \bmod p$, $\mathcal{T} = \prod_{j=1}^n \tau_j \bmod p$ and the following:

$$\beta = H(\mathcal{Z}, \mathcal{T}) \quad \text{and} \quad s_i = (k_i + \beta x_i) \bmod q \quad (4.2)$$

Then, $\sigma_i = (s_i, \beta)$ is the signature of message $\mathcal{Z}$ generated by member U_i . Finally, U_i broadcast

$\hat{X}_i$ and s_i to all other group members. It is noted that U_1 is assumed to be a trusted controller such that U_1 will be the last member to broadcast $\hat{X}_1$ and s_1 after all the other members U_j for $2 \leq j \leq n$ have broadcast $\hat{X}_j$ and s_j .

Authentication and Key Computation: After receiving the information from Round 2, U_i computes

$$\delta = \sum_{j=1}^n s_j \bmod q.$$

U_i then verifies the signatures by checking the following equation:

$$\beta = H(\mathcal{Z}, g^\delta \cdot (\prod_{j=1}^n y_j)^{-\beta}) \quad (4.3)$$

If equation (4.3) is correct, U_i will proceed to verify $\hat{X}_i$ that were broadcast in Round 2. However, if equation (4.3) is incorrect, it means that at least one of the signatures s_j in δ is incorrect. This might be due to an adversary trying to forge one or more of the signatures s_j for $j \leq n$. Then, all members will be requested to retransmit again. Alternatively, the trusted controller U_1 or any other group member can verify each signature by checking $\tau_i = g^{s_i} \cdot y_i^{-\beta}$ for $1 \leq i \leq n$. Once the incorrect signature $\sigma_k = (s_k, \beta)$ is detected, then the member U_k will be blocked from the GKA. This detection process should only be used if equation (4.3) fails as it requires quite a large amount of signature verifications if n is large.

From equation (4.1), we have $\hat{X}_i = g^{(r_i+c_i x_i)(r_{i+1}+c_{i+1} x_{i+1})-(r_{i-1}+c_{i-1} x_{i-1})(r_i+c_i x_i)} \bmod p$ for $1 \leq i \leq n$, where $\alpha_0 = \alpha_n$, $\alpha_{n+1} = \alpha_1$ and $\alpha = r, c$ and x . By checking $\prod_{j=1}^n \hat{X}_j = 1 \bmod p$, we can determine whether the $\hat{X}_i$ sent by U_i is genuine. If $\prod_{j=1}^n \hat{X}_j \neq 1 \bmod p$, it means that at least one of the $\hat{X}_j$ is incorrect. Then, all members will retransmit again. If each U_i correctly verifies that $\prod_{j=1}^n \hat{X}_j = 1 \bmod p$, then U_i computes the group key K as follows:

$$\begin{aligned} K &= (z_{i-1} \cdot y_{i-1}^{c_{i-1}})^{n(r_i+c_i x_i)} \cdot \hat{X}_i^{n-1} \cdot \hat{X}_{i+1}^{n-2} \cdots \hat{X}_{i-2} \bmod p \\ &= g^{(r_1+c_1 x_1)(r_2+c_2 x_2)+\cdots+(r_n+c_n x_n)(r_1+c_1 x_1)} \bmod p \end{aligned} \quad (4.4)$$

4.4.1 Security Analysis

In this section, we first discuss and prove that the proposed authenticated GKA protocol achieves fully authenticated GKA. As the proposed authenticated GKA protocol in Section 4.4 (without the signature information sent) looks similar to the BD protocol [BD95], we call it a variant of BD (VBD) protocol. We show that the VBD is secure if and only if the BD protocol is secure. We then prove that an adversary $\mathcal{A}$ is not able to obtain the group key K from the proposed authenticated GKA protocol and also prove that the proposed authenticated GKA protocol is secure against insider impersonation.

By requiring all group members to include their private key x_i for $1 \leq i \leq n$ in the final group key as shown in equation (4.4), the proposed protocol provides implicit key authentication. This is because the use of the private/public key pair by all group members provides proof that the final group key is known only to the parties taking part in the key agreement since only legitimate group members who knows their own private/public key can calculate the final group key. An adversary or non-group member without the knowledge of any of the private keys x_i will not be able to compute the final group key.

The generation of the signature s_i , together with the verification of all the signatures received as shown in equation (4.3) will provide evidence that all the received z_i in Round 1 are sent by genuine group members and not forged. An adversary will not be able to forge the signature as the Schnorr signature scheme is secure. The verification of the $\hat{X}_i$ by checking $\prod_{i=1}^n \hat{X}_i = 1 \bmod p$ will provide authentication for all $\hat{X}_i$ because the product of all $\hat{X}_i$ will not be equal to 1 if one of the $\hat{X}_i$ is forged. Without knowledge of a legitimate group member's private key x_i , a malicious adversary will not be able to forge any of the $\hat{X}_i$.

The combination of the verification of z_i using the Schnorr signature scheme, together with the verification of $\hat{X}_i$ using $\prod_{i=1}^n \hat{X}_i = 1 \bmod p$ provides a stronger form of authentication as it ensures each group member that the information received are authenticated and will result in the computation of the correct group key. Hence, we list the following theorem and give a formal proof.

Theorem 4 *The proposed authenticated GKA protocol achieves fully authenticated GKA with probability $1 - \epsilon\epsilon'$, where ϵ is the chance of forging the Schnorr signature scheme and ϵ' is the probability of computing the CDH problem.*

Proof: Assume that an adversary $\mathcal{A}$ wishes to impersonate a group member U_i (for some i), then $\mathcal{A}$ generates randoms $z_i = g^{r_i} \bmod p$ and $\tau_i = g^{k_i} \bmod p$ in Round 1. Then in Round 2, $\mathcal{A}$ has to compute the signature $\sigma_i = (s_i, \beta)$ where $s_i = (k_i + \beta x_i) \bmod q$ and $\beta = H(\prod_{j=1}^n z_j, \prod_{j=1}^n \tau_j)$. The validity of σ_i can be checked through the equation (4.3) by the other group members when they received all signatures. If equation (4.3) is incorrect, then the group member will know that some of the user is not a valid user. Now, assume that $\mathcal{A}$ is able to forge a Schnorr signature σ_i with probability ϵ , then $\mathcal{A}$ needs to produce a correct $\hat{X}_i = g^{(r_i + c_i x_i)(r_{i+1} + c_{i+1} x_{i+1}) - (r_{i-1} + c_{i-1} x_{i-1})(r_i + c_i x_i)} \bmod p$ in Round 2. As $\mathcal{A}$ does not know the private key x_i of U_i , hence, the probability of $\mathcal{A}$ being able to compute $\hat{X}_i$ is ϵ' as the CDH problem is difficult. Therefore, the chances of $\mathcal{A}$ impersonating a group member is $\epsilon\epsilon'$ which is very small as ϵ and ϵ' are negligible. Hence, the proposed authenticated GKA protocol achieves fully authenticated GKA with probability $1 - \epsilon\epsilon'$. ■

We shall now prove that the VBD protocol is secure as follows:

Theorem 5 *VBD protocol is secure if and only if BD protocol is secure.*

Proof:

($\implies$) We prove this by reduction proof, that is, for any instance protocol of BD, it can reduce to an instance of VBD. Now, consider a BD protocol as follows:

Round 1: Each user U_i ($1 \leq i \leq n$) chooses $a_i \in Z_q^*$ and computes $z_i = g^{a_i}$, send out z_i .

Round 2: Upon receipt of all z_i , each U_i ($1 \leq i \leq n$) computes $X_i = (z_{i+1}/z_{i-1})^{a_i}$, sends out X_i .

Computation of Key: $K = g^{a_n a_1 + \sum_{i=1}^{n-1} a_i a_{i+1}}$.

Now, we convert the above BD protocol into VBD protocol as follows: each user U_i ($1 \leq i \leq n$) chooses a random $r_i \in Z_q^*$ and computes $\bar{z}_i = g^{r_i}$ and its public key as $y_i = (z_i \bar{z}_i^{-1})^{1/c_i}$ where $c_i = H(\bar{z}_i)$. Then, it is easy to check that $a_i = r_i + c_i \log_g y_i$. Hence, by changing round 1 in the BD protocol as follows:

Round 1': Each user U_i ($1 \leq i \leq n$) sends out $\bar{z}_i = g^{r_i}$ and its public key y_i .

while round 2 and Computation of key are same as that of BD protocol respectively. Then, we obtain a VBD protocol. Hence, we conclude that BD protocol is secure if VBD protocol is secure.

($\Leftarrow$) We prove this by reduction proof, that is, for any instance of VBD, it can reduce to an instance of BD. Now, consider a VBD protocol as follows:

Round 1: Each user U_i ($1 \leq i \leq n$) has public/private key ($y_i = g^{x_i}, x_i$). First, U_i chooses $r_i \in Z_q^*$ and computes $z_i = g^{r_i}$, sends out z_i and y_i .

Round 2: Upon receipt of all z_j ($1 \leq j \leq n$), each U_i ($1 \leq i \leq n$) computes

$$X_i = \left(\frac{z_{i+1} \cdot y_{i+1}^{c_{i+1}}}{z_{i-1} \cdot y_{i-1}^{c_{i-1}}} \right)^{r_i + c_i x_i}$$

where $c_j = H(z_j)$ for $j = i, i-1, i+1$ and sends out X_i .

Computation of Key: $K = g^{(r_n + c_n x_n)(r_1 + c_1 x_1) + \sum_{i=1}^{n-1} (r_i + c_i x_i)(r_{i+1} + c_{i+1} x_{i+1})}$.

Now, we convert the above VBD protocol into BD protocol as follows: each user U_i ($1 \leq i \leq n$), set $\bar{z}_i = z_i \cdot y_i^{c_i}$ and $a_i = r_i + c_i x_i \bmod q$ where $c_i = H(z_i)$. Then, changing round 1 and round 2 in VBD as follows:

Round 1': Each user U_i ($1 \leq i \leq n$) sends out $\bar{z}_i$.

Round 2': Upon receipt of all $\bar{z}_i$, each U_i ($1 \leq i \leq n$) computes $\bar{X}_i = (\bar{z}_{i+1} / \bar{z}_{i-1})^{a_i}$ and sends $\bar{X}_i$.

Computation of Key: $K = g^{a_n a_1 + \sum_{i=1}^{n-1} a_i a_{i+1}}$.

Then, we obtain a BD protocol. Hence, VBD protocol is secure if BD protocol is secure. $\blacksquare$

Next, we prove that an adversary $\mathcal{A}$ is not able to obtain the group key K from the proposed authenticated GKA protocol.

Theorem 6 *An adversary $\mathcal{A}$ is unable to obtain the group key K in the proposed authenticated GKA protocol provided the CDH problem is difficult and the Schnorr signature scheme and the VBD protocol are secure.*

Proof: First, consider that an adversary $\mathcal{A}$ tries to impersonate a group member U_i (for some i) to obtain a group key. But by Theorem 4, it is shown that the adversary $\mathcal{A}$ is not able to impersonate a group member as the Schnorr signature scheme is secure and the CDH problem is difficult. Hence, the adversary $\mathcal{A}$ is not able to impersonate the user U_i to establish a group key. Now, consider that an adversary $\mathcal{A}$ can only obtain the public information from the group members

Table 4.1: Complexity Analysis

Methods	No. of Rounds	No. of Signature	No. of sign. verification
1: BD with digital signatures	2	2	$n + 1$
2: BD with extra round	3	2	$n + 1$
3: proposed protocol	2	1	1

sent. Then, this is just the VBD protocol by ignoring the signature (σ_j for $1 \leq j \leq n$) in round 2 and τ_j (for $1 \leq j \leq n$) in round 1. By Theorem 5, it is shown that the VBD protocol is secure. Therefore, the adversary $\mathcal{A}$ is not able to obtain the group key established by a group. ■

Next, we prove that the proposed protocol is secure against insider impersonation.

Theorem 7 *The proposed authenticated GKA protocol is secure against insider impersonation if the Schnorr signature scheme is secure.*

Proof: Assume that there is k ($< n - 1$) number of group member in a group of n members to impersonate a user U_i (for some i) to establish a new group key, then there is at least one group member, say U_j (who is not one of the k impersonators) sending a new z'_j in round 1 which is different from that of the previously sent. Therefore, $\mathcal{Z}' = \prod_{k=1}^n z'_k$ will be different from the previous $\mathcal{Z}$ sent. In this case, the k number of impersonators need to produce a valid signature σ'_i (which makes a commitment of τ_i in round 1) of the message $\mathcal{Z}'$ for the user U_i in round 2. But, this is impossible for the k number of impersonators to produce such valid signature as they do not know the private key x_i of U_i and the Schnorr signature scheme is secure. Hence, it is not possible for any k ($< n - 1$) insider to impersonate any group member to establish a new group key. ■

4.5 Comparison of Computational Complexity

In Table 4.1, we compare the computational complexity of the proposed protocol (called Method 3) with the intuitive method of using digital signatures (Method 1) and the use of an extra round (Method 2) to provide authentication for the BD.

The first method is the intuitive method of authenticating the BD using digital signatures and is described in Section 4.3.1. Each group member U_i is required to generate two signatures, one

for authenticating z_i and the other for $\hat{X}_i$. As only z_{i+1} and z_{i-1} are required for computing $\hat{X}_i$ in Round 2, each U_i would only have to verify these two terms received from Round 1. Since all $\hat{X}_i$ are used for the computation of the group key, each member has to verify $n - 1$ $\hat{X}_i$, where n is the group size. This results in $n + 1$ signature verifications for each U_i as shown in Table 4.1. The advantage of this method is that there is no increase in the number of communication rounds.

The second method involves authenticating the BD by adding an extra round of communications and is described in Section 4.3.2. This method has also been used in papers [KY03] and [NLKW04] to provide authentication. An advantage of using this method is that it provides a modular approach that simplifies the design and analysis of the authenticated GKA protocol. However, this method requires an *extra* round of communication. Like the first method, this method requires each U_i to generate two signatures and perform $n + 1$ signature verifications as shown in Table 4.1. However, the signature verification of this method is different from the first method as the instance identifiers (IIDs) ϕ_i that are stored by each U_i are used in the signature verification.

The proposed protocol, on the other hand, does not increase the number of communication rounds like the first method. Furthermore, the proposed protocol is more efficient as it requires only *one* signature generation and only *one* signature verification to be performed by each member regardless of the group size as shown in Table 4.1. This would result in large savings in computational cost as compared to the first two methods.

4.6 Summary

In this chapter, we first examined the BD protocol in detail and described a unique characteristic of the BD. Next, we presented techniques currently used to provide authentication for the BD. We then proposed an efficient authenticated GKA protocol that requires each group member to sign only *one* message and perform only *one* signature verification, regardless of the group size. The proposed protocol also does not require any additional communication rounds. The security analysis shows that the proposed protocol is secure against malicious adversaries and also insider impersonations where k ($< n - 1$) malicious group members collaborate to masquerade as a honest node. In the complexity analysis, we compared the proposed protocol with two other methods of providing authentication for the BD. We showed that the proposed protocol is more efficient.

Chapter 5

Authenticated Dynamic Group Key Agreement for Autoconfigurable MANETs

5.1 Introduction

Symmetric wireless networks such as mobile ad hoc networks (MANETs) have higher dynamic network topology than wired networks because user nodes in such wireless networks are generally mobile and battery-operated such as PDAs and laptop computers as compared to user nodes in wired networks which are typically desktop systems that have no mobility. For example, wireless nodes may move out of connectivity or run out of batteries, thus resulting in a user leave event. Wireless nodes may also switch off their devices and join the secure group communications at a later time, resulting in a Join event. Hence, dynamic GKA protocols such as Join and Leave Protocol are necessary to ensure that the session key is updated whenever users join or leave the group to provide backward secrecy (i.e. new users cannot discover previous keys used) and forward secrecy (i.e. leaving users cannot discover new keys used).

Another characteristic of MANETs is that MANET nodes have routing capabilities to forward messages for other nodes. However, before any proper routing of messages can occur, all nodes must be configured with unique network layer (IP) addresses. Since MANETs are self-organizing and have dynamic network topology, the IP address assignment should be done automatically and dynamically. Hence, besides handling dynamic events, GKA protocols for wireless networks such as MANETs should also address issues such as IP address assignment in the network.

In [BPS⁺04], the authors proposed a secure autoconfiguration protocol for MANET that uses threshold cryptography techniques. However, that paper did not describe network merging techniques in detail, which was mentioned in [WZ04] to be a significant challenge for MANET autoconfiguration. Furthermore, that paper focused only on securing IP address autoconfiguration and does not describe GKA for secure group communication.

The most intuitive method of providing GKA with secure IP address autoconfiguration will be to run a secure autoconfiguration protocol first followed by a GKA protocol. However, this method will result in a waste of energy for MANET nodes that have limited power capabilities as they will have to perform communication and computation twice.

In the previous chapter, we have seen that the BD protocol is suitable for GKA in wireless networks. However, the BD protocol is a static protocol, that is, it does not take care of dynamic events such as user join, user leave and network merge. There have been several research work done to extend the BD protocol to handle dynamic events. In [STW98], Steiner et al briefly suggested a Join Protocol for adding a new member for the BD protocol while in [DB05], Dutta and Barua proposed provably secure constant round dynamic authenticated GKA protocols (Join and Leave Protocols) based on the BD for dynamic scenario. Dutta and Barua's Leave protocol requires only the left and right neighbors of leaving users to change their random secrets.

In the first part of this chapter, we show that the suggested Join method for BD by Steiner et al in [STW98] does not provide backward secrecy (new users can discover previous keys used). Next, we show that Dutta and Barua's Leave protocol in [DB05] does not achieve forward secrecy (a leaving user can discover the new session key used in subsequent sessions if any two consecutive remaining users do not change their random secrets).

In the second part of this chapter, we propose four efficient protocols, namely the Initial key agreement, Join, Leave and Merge protocols, that provide both authenticated GKA and secure IP address autoconfiguration simultaneously for symmetric wireless networks such as MANETs where all nodes have equal limited computing capabilities. As compared to existing protocols such as [BPS⁺04], our Merge protocol is able to resolve this address conflict issue in a more systematic, effective and highly efficient way. The dynamic protocols, namely Join, Leave and Merge protocols, also provide effective and efficient solutions to the problem of frequent network topology

changes in MANET by changing the group key and updating the IP address allocation table simultaneously whenever a dynamic event occurs. Our dynamic protocols also reduce computational and communication costs for the nodes as only certain nodes perform exponentiations in the Join and Merge Protocols and only half the group broadcast messages in the first round of the Leave Protocol.

Our proposed protocols take advantage of the broadcast nature of MANET as compared to GDH where the mode of communication is unicast and each user U_i has to wait until he has received from the user before him U_{i-1} before he can compute and transmit to the next user U_{i+1} . Furthermore, our protocols use a so-called ring structure in GKA where the users are arranged in a ring while TGDH and STR uses the tree structure, which contains not only the MANET users but also intermediate nodes that are parent nodes of two child nodes. As compared to the tree structure, the ring structure is less complicated and can be easily converted into a list to be used together with the IP address allocation table. During dynamic events such as join, leave and merge, it is more complicated to update the tree structure used in TGDH and STR as compared to the ring structure due to the use of intermediate nodes in TGDH and STR. In the complexity analysis and discussion section, we show that despite having additional capabilities, our protocols are still comparatively efficient.

5.2 Cryptanalysis of Steiner et al. Join Protocol for BD

In [STW98], Steiner et al described a method for adding a new member for the BD protocol as follows:

Suppose a new member U_{n+1} wants to join the group. First, U_n chooses a random r'_n and broadcast $g^{r'_n} \bmod p$. Next, U_{n+1} chooses a random r_{n+1} and broadcast $g^{r_{n+1}} \bmod p$. Subsequently, U_1, U_{n+1} ,

U_n and U_{n-1} compute and broadcast the following:

$$\begin{aligned} X'_1 &= g^{r_1 r_2 - r_1 r_{n+1}} \bmod p, \\ X'_{n+1} &= g^{r_{n+1} r_1 - r'_n r_{n+1}} \bmod p, \\ X'_n &= g^{r'_n r_{n+1} - r_{n-1} r'_n} \bmod p, \\ X'_{n-1} &= g^{r_{n-1} r'_n - r_{n-1} r_{n-2}} \bmod p \end{aligned}$$

respectively. All other X_i for $2 \leq i \leq n-2$ are taken from the previous run of the BD protocol. Finally, all group members, including U_{n+1} compute the new group key in the usual BD approach as described in Section 2.7.6, where the new group key, K' is:

$$K' = g^{r_1 r_2 + r_2 r_3 + \dots + r_{n-1} r'_n + r'_n r_{n+1} + r_{n+1} r_1} \bmod p.$$

However this method does not provide backward secrecy as the new member U_{n+1} can recover the previous group key shared by the group member U_i for $1 \leq i \leq n$. We show this as follows (for simplicity, “mod p ” is omitted from the expressions if no confusion arises): As U_{n+1} knows all $\bar{X}_i$ for $1 \leq i \leq n+1$ where

$$\bar{X}_i = \begin{cases} X_i & \text{if } 2 \leq i \leq n-2, \\ X'_i & \text{if } i = 1, n, n-1, n+1. \end{cases}$$

The member U_{n+1} first computes $g^{r_{n+1} r_1} = \bar{X}_{n+1} \cdot g^{r'_n r_{n+1}}$ and recursively compute $g^{\bar{r}_j \bar{r}_{j+1}} = \bar{X}_j \cdot g^{\bar{r}_{j-1} \bar{r}_j}$ for $1 \leq j \leq n+1$ where $\bar{r}_{n+2} = r_1$, $\bar{r}_n = r'_n$, $\bar{r}_{n+1} = r_{n+1}$ and $\bar{r}_i = r_i$ for $1 \leq i \leq n-1$. Hence, U_{n+1} knows $g^{r_j r_{j+1}}$ for $1 \leq j \leq n-2$ so U_{n+1} would obtain $g^{r_n r_1}$ and $g^{r_{n-1} r_n}$ from $X_1 = g^{r_1 r_2 - r_n r_1}$ and $X_{n-1} = g^{r_{n-1} r_n - r_{n-2} r_{n-1}}$ respectively as U_{n+1} knows $g^{r_1 r_2}$ and $g^{r_{n-2} r_{n-1}}$. As such, U_{n+1} can obtain the previous group key as $K = g^{r_n r_1} \cdot \prod_{j=1}^{n-1} g^{r_j r_{j+1}}$. Therefore, Steiner et al. method does not provide backward secrecy in the group key agreement.

5.3 Cryptanalysis of Dutta and Barua's Leave Protocol

In [DB05], Dutta and Barua extended the work of Burmester and Desmedt [BD95] and proposed provably secure constant round authenticated GKA protocols in dynamic scenario, where users can join or leave the group at their desire and the session key is updated. They also provided security proofs for their dynamic authenticated GKA, namely Join and Leave Protocol. Dutta and Barua's Leave protocol as described in Section 2.7.12 requires only the left and right neighbors of leaving users to change their random secrets. We shall show that their Leave protocol does not achieve forward secrecy (a leaving user can still discover the new session key used in subsequent sessions if any two consecutive remaining users do not change their random secret).

5.3.1 Attack on the Dutta and Barua's Leave Protocol

In Dutta and Barua's Leave Protocol as mentioned in Section 2.7.12, we have a group of m users $\mathcal{L} = \{U_{l_1}, \dots, U_{l_m}\}$ leaving the group $\mathcal{G} = \{U_1, \dots, U_n\}$, who has already established a session key in the previous group key agreement. Then the new group of users are $\{U_1, \dots, U_{l_1}^L\} \cup \{U_{l_1}^R, \dots, U_{l_2}^L\} \cup \dots \cup \{U_{l_m}^R, \dots, U_n\}$, where $U_{l_i}^L$ and $U_{l_i}^R$ are respectively the left and right neighbors of leaving user U_{l_i} for $1 \leq i \leq m$. Then for any leaving user U_{l_i} , $U_{l_i}^L = U_{l_i-k}$ if the k consecutive users $U_{l_i}, U_{l_i-1}, \dots, U_{l_i-(k-1)}$ are all leaving and U_{l_i-k} is not leaving the group. Similarly, $U_{l_i}^R = U_{l_i+k'}$ if the k' consecutive users $U_{l_i}, U_{l_i+1}, \dots, U_{l_i+(k'-1)}$ are all leaving and $U_{l_i+k'}$ is not leaving the group. The $n - m$ remaining users are re-indexed as $\mathcal{G}' = \{V_1, \dots, V_{n-m}\}$. The left and right keys of the remaining users are also re-indexed and denoted by two arrays $\{\hat{K}_1^L, \dots, \hat{K}_{n-m}^L\}$ and $\{\hat{K}_1^R, \dots, \hat{K}_{n-m}^R\}$ respectively.

Whenever users leave the group, the session key has to be changed by the Leave Protocol to prevent leaving users from discovering the new session key used, thus providing forward secrecy. Let $U_{j_L} = U_{l_i}^L$ and $U_{j_R} = U_{l_i}^R$ be the left and right neighbors respectively of a leaving user U_{l_i} . Dutta and Barua's Leave Protocol requires only the left neighbor U_{j_L} and right neighbor U_{j_R} of a leaving user U_{l_i} to choose new random secrets $\hat{r}_{j_L} \in Z_q^*$ and $\hat{r}_{j_R} \in Z_q^*$ respectively for computation of the new session key. However, we show here that if any two consecutive remaining users, say U_γ and $U_{\gamma+1}$, do not change their random secrets r_γ and $r_{\gamma+1}$ respectively, then Dutta and Barua's Leave Protocol will not provide forward secrecy. The attack method is as follows:

Without loss of generality, we assume that the leaving user be U_δ for some $\delta > 1$ and users

$U_{\delta+1}, U_{\delta+2}, U_{\delta+3}$ and $U_{\delta+4}$ do not leave the group. By the Leave Protocol as shown in Section 2.7.12, users $U_{\delta+2}$ and $U_{\delta+3}$ will not change their random secret element $r_{\delta+2}$ and $r_{\delta+3}$ respectively. Then, the leaving user U_{δ} knows $g^{r_{\delta+2}r_{\delta+3}}$ from the previous key computation and also knows $\hat{Y}_k = \hat{K}_k^R / \hat{K}_k^L = g^{\hat{r}_k \hat{r}_{k+1} - \hat{r}_{k-1} \hat{r}_k}$ for $1 \leq k \leq n - m$ where $\hat{r}_{n-m+1} = \hat{r}_1$ and $\hat{r}_0 = \hat{r}_{n-m}$. From the observation of Lemma 1 in Section 2.7.12, the user $U_{\delta+2}$ is able to compute $g^{\hat{r}_k \hat{r}_{k+1}}$ for $1 \leq k \leq n - m$ where $\hat{r}_{n-m+1} = \hat{r}_1$, $\hat{r}_{\delta+2} = r_{\delta+2}$ and $\hat{r}_{\delta+3} = r_{\delta+3}$ by knowing $g^{r_{\delta+2}r_{\delta+3}}$. Therefore, by similar computations, the leaving user U_{δ} is also able to compute all $g^{\hat{r}_k \hat{r}_{k+1}}$ for $1 \leq k \leq n - m$, where $\hat{r}_{n-m+1} = \hat{r}_1$, $\hat{r}_{\delta+2} = r_{\delta+2}$ and $\hat{r}_{\delta+3} = r_{\delta+3}$. Hence, leaving user U_{δ} can compute the new session key sk where

$$sk' = \prod_{k=1}^{n-m} g^{\hat{r}_k \hat{r}_{k+1}} \bmod p$$

and $\hat{r}_{n-m+1} = \hat{r}_1$. Therefore, Dutta and Barua's Leave Protocol does not provide forward secrecy if any two consecutive remaining users do not change their random secrets.

5.3.2 Flaw in Dutta and Barua's security proof

In the Leave Protocol by Dutta and Barua in Section 2.7.12, some of the left and right keys that are currently used by remaining users were created and used in a previous GKA protocol run. However, the re-use of these information (left and right keys that were created in earlier GKA protocol run and are currently being used by remaining users) in the Leave Protocol was not included in the security analysis because the security analysis does not take care of leaving users who know additional information.

As a result, Dutta and Barua's security proof fails to consider the ability of a leaving user, who is no longer part of the group, being able to obtain the new group key by making use of the re-use information (i.e. left and right keys that were created in earlier GKA protocol run and are currently being used by remaining users) and using the technique mentioned in Section 5.3.1.

5.4 Authenticated Dynamic GKA for Autoconfigurable MANETs

Before describing the proposed protocols, we first give an introduction of the different classes of IP address autoconfiguration and show the requirements (including security requirements) for GKA protocols to be applied to MANETs.

5.4.1 IP Address Autoconfiguration Classes

IP address autoconfiguration protocols can be classified into three types of approaches: Stateful, stateless and hybrid protocol. Stateful approaches such as [BPS⁺04] assumes the existence of a central entity to assign unique addresses to unconfigured nodes and keep state information about currently assigned addresses in an address allocation table. Stateful approaches are further distinguished by three categories: *Centralized maintenance of allocation table*, *distributed maintenance of allocation table*, *distributed maintenance of multiple disjoint allocation tables* approaches. The stateless approach does not maintain any allocation table. Instead, an unconfigured node self-assigns an address and verifies its uniqueness in a distributed manner using the Duplicated Address Detection (DAD) procedure [PMW⁺01]. The hybrid approach [Wen05], combines elements from both stateful and stateless approaches, leading to more robust protocols.

5.4.2 Requirements

Several requirements must be satisfied before we can apply our protocol in MANET. First, we require that a Public Key Infrastructure (PKI) should be in place. All MANET nodes taking part in the secure group communication will first generate their private-public key pair and register with the Certification Authority (CA) to certify their public key and obtain a public key certificate through a secure channel before taking part in group communications. The CA could simply be the owner of the MANET.

As the network topology of MANET is dynamic, one requirement for GKA in MANET will be to efficiently handle dynamic events. Furthermore, to prevent address conflicts that are caused by network merging from affecting the routing of data packets, the IP autoconfiguration protocol

should resolve these address conflicts as quickly as possible.

To handle synchronization and delay issues, each MANET user is equipped with a timeout mechanism. Whenever user U_i for $1 \leq i \leq n$ broadcast a message, all other users will activate the timeout mechanism. The timeout interval t_{out} can be computed based on the worst case delay to receive a message broadcast from the furthest MANET user or based on other statistics. If a user does not receive the broadcast message after timeout t_{out} , then the user will send a ReTransmission reQuest (RTQ). If a node U_a is still unable to receive the broadcast message after r number of RTQ requests, then the group will continue the GKA without U_a . U_a can rejoin the group later with a join request. Note that the value r can be selected depending on the implementation. For example, to have shorter delay during GKA, then r is set to be very small, e.g. $r = 3$.

We also require that only trusted and honest group members/nodes can participate in the GKA and IP address autoconfiguration service. The messages sent during the GKA must be authenticated using digital signatures. The GKA protocol must also generate a key that is fresh whenever dynamic events occur.

5.5 The Proposed Protocols

Our proposed protocols provide both secure IP address autoconfiguration and authenticated GKA simultaneously to produce an efficient solution of securing MANET communications. In this section, we describe an authenticated initial key agreement (IKA) protocol and three authenticated dynamic protocols, namely the Join, Leave and Merge protocol, which are more efficient than re-executing the BD protocol whenever dynamic events occur. Although any signature scheme can be used in our proposed protocols, we choose the efficient Rabin's signature scheme described in Section 2.3.4 as it requires only modulo multiplication which has lower computational costs than modulo exponentiations for signature verification. Furthermore, our proposed dynamic protocols use symmetric key cryptography that have been shown to have energy requirements of orders of magnitude lower than modular exponentiations [CKM00, Mir].

Let p (1024 bits) and q (160 bits) be prime numbers such that q divides $p-1$ and H be a one-way hash function. Next, an element $g \in Z_p^*$ of order q is selected. Each user U_i for $1 \leq i \leq n$ will keep its private key $x_i \in Z_q^*$ and two random large (512 bits) prime numbers $\bar{p}_i$ and $\bar{q}_i$ in secret (where

$(\bar{p}_i, \bar{q}_i)$ is U_i 's Rabin's signature scheme private key) and publish its public key $y_i = g^{x_i} \bmod p$ and Rabin's signature scheme public key $\bar{N}_i = \bar{p}_i \cdot \bar{q}_i$. The certificate $\text{cert}[U_i]$ for user U_i comprises of the user identity U_i , public key y_i , Rabin signature public key $\bar{N}_i$ and a Rabin signature signed by the Certification Authority (CA) to certify that both y_i and $\bar{N}_i$ belongs to user U_i . The IP address of U_i is denoted as $\text{addr}[U_i]$. We describe the four proposed protocols in detail below.

5.5.1 Initial Key Agreement (Group Formation) Protocol

Let $\mathcal{G} = \{U_1, \dots, U_n\}$ be the initial group of n users. We consider a so called ring structure among the users of $\mathcal{G}$ where the users' indices can be considered on the circulation of $\{1, \dots, n\}$. We also assume that the IP addresses $\text{addr}[U_i]$ of $U_i \in \mathcal{G}$ for $1 \leq i \leq n$ belong to the same subnet. For example, the IP address $\text{addr}[U_i]$ of $U_i \in \mathcal{G}$ for $1 \leq i \leq n$ will begin with $(179.156.0.i)$ and hence do not have address conflicts. The Initial Key Agreement (IKA) protocol consists of two rounds during which each user U_i will broadcast their IP address $\text{addr}[U_i]$ and keying materials for group key computation. At the end of the second round, all users $U_i \in \mathcal{G}$ will be able to compute the common group key K as well as obtain the IP addresses of all other users which will be filled in an allocation table LIST. The Initial Key Agreement (IKA) Protocol is described below:

Round 1: Each user U_i for $1 \leq i \leq n$ first selects $r_i \in Z_q^*$ and computes $z_i = g^{r_i} \bmod p$. U_i then broadcasts $\{U_i || z_i || \text{addr}[U_i] || \text{cert}[U_i]\}$.

Round 2: Each U_i for $1 \leq i \leq n$ first verifies the $\text{cert}[U_j]$ of other users U_j for $1 \leq j \neq i \leq n$. If the verification is not correct, U_i rejects U_j . Otherwise, U_i stores the IP address $\text{addr}[U_j]$ of all user U_j for $1 \leq j \neq i \leq n$ in the IP allocation table LIST. Then user U_i computes $c_\alpha = H(z_\alpha)$ for $\alpha = i, i-1, i+1$ where $c_0 = c_n$ and $c_{n+1} = c_1$ and

$$\hat{X}_i = \left(\frac{z_{i+1} \cdot y_{i+1}^{c_{i+1}}}{z_{i-1} \cdot y_{i-1}^{c_{i-1}}} \right)^{d_i} \bmod p \quad (5.1)$$

where $d_i = r_i + c_i x_i$. Finally, each U_i computes $Z = \prod_{j=1}^n z_j \bmod p$ and signs $U_i || z_i || \hat{X}_i || Z || \text{addr}[U_i]$ to obtain the Rabin's signature (σ_i, τ_i) and broadcasts $\{U_i || \hat{X}_i || \text{addr}[U_i] || (\sigma_i, \tau_i)\}$. All users then store Z for verification in the next round. It is noted that U_1 is assumed to be a trusted controller such that U_1 will be the last user to broadcast its message after all the other users U_j for $2 \leq j \leq n$

have broadcast their messages.

Authentication and Key Computation: Each U_i for $1 \leq i \leq n$ verifies the Rabin's signature (σ_j, τ_j) of message $U_j || z_j || \hat{X}_j || \mathcal{Z} || \text{addr}[U_j]$ from U_j for $1 \leq j \neq i \leq n$, where $\mathcal{Z}$ was stored in Round 2. If the signature verifications are correct, U_i computes the group key K . From equation (5.1), we have $\hat{X}_i = g^{d_i d_{i+1} - d_{i-1} d_i} \bmod p$ where $d_i = r_i + c_i x_i$ for $1 \leq i \leq n$ and $d_0 = d_n$ and $d_{n+1} = d_1$. Then U_i can compute $g^{d_i d_{i+1}} = \hat{X}_i \cdot g^{d_{i-1} d_i} \bmod p$ and recursively compute $g^{d_j d_{j+1}}$ using similar method as described in Lemma 3 case (i) and case (ii), where $d_j = r_j + c_j x_j$ for $1 \leq j \neq i \leq n$ and $d_{n+1} = d_1$. Hence, each U_i computes the group key K as follows:

$$K = \prod_{j=1}^n g^{d_j d_{j+1}} = g^{d_1 d_2 + \dots + d_n d_1} \bmod p \quad (5.2)$$

where $d_j = r_j + c_j x_j$ for $1 \leq j \leq n$ and $d_0 = d_n$ and $d_{n+1} = d_1$. However, if U_i detects that the signature (σ_a, τ_a) is incorrect, then user U_a will be excluded from the GKA. U_{a-1} will recompute $\hat{X}'_{a-1} = g^{d_{a-1} d_{a+1} - d_{a-2} d_{a-1}} \bmod p$ and U_{a+1} will recompute $\hat{X}'_{a+1} = g^{d_{a+1} d_{a+2} - d_{a-1} d_{a+1}} \bmod p$ respectively where $d_\alpha = r_\alpha + c_\alpha x_\alpha$ and $\alpha = a-2, a-1, a+1$ and $a+2$. U_{a-1} and U_{a+1} then sign $U_{a-1} || \hat{X}'_{a-1}$ and $U_{a+1} || \hat{X}'_{a+1}$ to obtain $(\sigma'_{a-1}, \tau'_{a-1})$ and $(\sigma'_{a+1}, \tau'_{a+1})$ respectively. Finally U_{a-1} broadcasts $\{U_{a-1} || \hat{X}'_{a-1} || (\sigma'_{a-1}, \tau'_{a-1})\}$ and U_{a+1} broadcasts $\{U_{a+1} || \hat{X}'_{a+1} || (\sigma'_{a+1}, \tau'_{a+1})\}$. All users will exclude U_a from GKA by using new $\hat{X}'_{a-1}$ and $\hat{X}'_{a+1}$ and removing $\text{addr}[U_a]$ from LIST.

5.5.2 Join Protocol

Let $\mathcal{G} = \{U_1, \dots, U_n\}$ be the original group and U_{n+1} be the new user joining the group. We divide $\mathcal{G}$ into two parts, $\{U_1, U_n\}$, which are the two users actively involved in the Join Protocol and $\{U_2, \dots, U_{n-1}\}$, which consist of the rest of the group. We consider a ring structure among the users of $\mathcal{G}$ with U_{n+1} joining $\mathcal{G}$ in between U_n and U_1 to form the new group $\mathcal{G}' = \{U_1, \dots, U_{n+1}\}$. To initiate the Join Protocol, U_{n+1} first selects a temporary IP address from a special range of IP addresses for temporary allocation and broadcast its `join_request` message.

The Join Protocol consists of three rounds during which U_n will assign an IP address to new user U_{n+1} . U_n will also send the new IP address allocation table LIST' and new keying materials to U_{n+1} securely by establishing a secure channel using authenticated Diffie-Hellman (DH) key exchange.

Both U_1 and U_n will also compute and broadcast new keying materials securely to $\mathcal{G}$ by encrypting them with the previously established group key K . Finally, all current users in $\mathcal{G}$ update their IP address allocation table to $LIST'$ by adding U_{n+1} 's IP address to it and all users including U_{n+1} in new group $\mathcal{G}'$ will compute the new group key K' . The Join Protocol is described in detail below:

Round 1:

(i) The new node U_{n+1} first chooses a random $r_{n+1} \in Z_q^*$ and computes $z_{n+1} = g^{r_{n+1}} \bmod p$. Next, U_{n+1} signs $U_{n+1}||z_{n+1}$ to obtain the Rabin's signature $(\sigma_{n+1}, \tau_{n+1})$. Finally, U_{n+1} broadcasts $\{U_{n+1}||z_{n+1}||(\sigma_{n+1}, \tau_{n+1})||\mathbf{cert}[U_{n+1}]\}$.

(ii) U_n selects an IP address $\mathbf{addr}[U_{n+1}]$ that is not in $LIST$ for U_{n+1} . Next, U_n signs $U_n||z_n||\mathbf{addr}[U_{n+1}]$ to obtain Rabin's signature (σ'_n, τ'_n) and broadcasts $\{U_n||z_n||\mathbf{addr}[U_{n+1}]||(\sigma'_n, \tau'_n)||\mathbf{cert}[U_n]\}$.

Round 2: U_1 first verifies the correctness of the $\mathbf{cert}[U_{n+1}]$ and signature $(\sigma_{n+1}, \tau_{n+1})$ from U_{n+1} . If they are invalid, then U_1 rejects U_{n+1} . Otherwise, U_1 chooses a new random $r'_1 \in Z_q^*$ and computes K^* as follows:

$$\begin{aligned} K^* &= K \cdot (z_2 \cdot y_2^{c_2} \cdot z_n \cdot y_n^{c_n})^{-d_1} \cdot (z_2 \cdot y_2^{c_2} \cdot z_{n+1} \cdot y_{n+1}^{c_{n+1}})^{d'_1} \\ &= g^{d'_1 d_2 + d_2 d_3 + \dots + d_{n-1} d_n + d_{n+1} d'_1} \bmod p. \end{aligned} \quad (5.3)$$

where $d_i = r_i + c_i x_i$ and $c_i = H(z_i)$ for $2 \leq i \leq n$ and $d'_1 = r'_1 + c'_1 x_1$ and $c'_1 = H(g^{r'_1})$. U_1 then encrypts $K^*||U_1$ using previously established group key K and a symmetric key encryption SKE to obtain $c_1 = \mathcal{E}_K(K^*||U_1)$ and broadcasts $\{U_1||c_1\}$.

Round 3 : U_n verifies the $\mathbf{cert}[U_{n+1}]$ and signature $(\sigma_{n+1}, \tau_{n+1})$ from U_{n+1} . If they are invalid, then U_n rejects U_{n+1} . Otherwise, U_n computes the DH key $K_{U_n U_{n+1}} = (z_{n+1} \cdot y_{n+1}^{c_{n+1}})^{d_n} = g^{d_n d_{n+1}} \bmod p$ that it shares with U_{n+1} , where $d_k = r_k + c_k x_k$ for $k = n$ and $n + 1$. Next, U_n encrypts $K_{U_n U_{n+1}}||\mathbf{addr}[U_{n+1}]||U_n$ using the previously established group key K and obtains $c_n = \mathcal{E}_K(K_{U_n U_{n+1}}||\mathbf{addr}[U_{n+1}]||U_n)$. U_n then decrypts c_1 and obtains K^* and U_1 . If the identity U_1 is not decrypted correctly, then U_n rejects c_1 . Otherwise, U_n updates $LIST$ to $LIST'$ by including the IP address $\mathbf{addr}[U_{n+1}]$ of the new user U_{n+1} and encrypts $K^*||LIST'||U_n$ using $K_{U_n U_{n+1}}$ and obtains $c'_n = \mathcal{E}_{K_{U_n U_{n+1}}}(K^*||LIST'||U_n)$. Finally, U_n broadcast $\{U_n||c_n||c'_n\}$.

Key Computation :

- (i) U_{n+1} first verifies the $\text{cert}[U_n]$ and signature (σ'_n, τ'_n) from U_n . If they are invalid, then U_{n+1} rejects U_n . Otherwise, U_{n+1} sets its IP address to $\text{addr}[U_{n+1}]$ and discards the temporary IP address used. U_{n+1} then computes $K_{U_n U_{n+1}} = (z_n \cdot y_n^{c_n})^{d_{n+1}} = g^{d_n d_{n+1}} \bmod p$ where $d_k = r_k + c_k x_k$ for $k = n$ and $n + 1$. Next, U_{n+1} decrypts c'_n and obtains K^* , LIST' and U_n respectively. If the identity U_n is not decrypted correctly, then U_{n+1} rejects c'_n . Otherwise, U_{n+1} accepts K^* and stores LIST' .
- (ii) Users U_i for $2 \leq i \leq n - 1$ decrypt c_1 using previously established group key K and obtain K^* and identity U_1 . If the identity U_1 is not decrypted correctly, then U_i rejects c_1 . Otherwise, U_i accepts K^* . Next, users U_i for $1 \leq i \leq n - 1$ decrypts c_n using previously established group key K and obtain $K_{U_n U_{n+1}}$, $\text{addr}[U_{n+1}]$ and identity U_n . If the identity U_n is not decrypted correctly, then U_i rejects c_n . Otherwise, U_i accepts $K_{U_n U_{n+1}}$ and updates LIST to LIST' by including the IP address $\text{addr}[U_{n+1}]$ of the new user U_{n+1} and proceed to compute the new group key K' .
- (iii) Finally, all users U_i for $1 \leq i \leq n$ and U_{n+1} compute the new group key K' as follows:

$$\begin{aligned} K' &= K^* \cdot K_{U_n U_{n+1}} \\ &= g^{d'_1 d_2 + d_2 d_3 + \dots + d_n d_{n+1} + d_{n+1} d'_1} \bmod p \end{aligned} \quad (5.4)$$

where $d_i = r_i + c_i x_i$ and $c_i = H(z_i)$ for $2 \leq i \leq n + 1$ and $d'_1 = r'_1 + c'_1 x_1$ and $c'_1 = H(g^{r'_1})$.

5.5.3 Merge Protocol

A merge occurs when two or more subgroups are combined into a single group; it can also be seen as multiple users joining a group. Let $\mathcal{G}_A = \{U_1, \dots, U_n\}$ denote all users in Group $\mathcal{G}_A$ with previously established group key K_A and IP address allocation table LIST_A and $\mathcal{G}_B = \{U_{n+1}, U_{n+2}, \dots, U_{n+m}\}$ denote all users in Group $\mathcal{G}_B$ with previously established group key K_B and IP address allocation table LIST_B . We divide $\mathcal{G}_A$ into two parts, $\{U_1, U_n\}$ and $\{U_2, \dots, U_{n-1}\}$ and $\mathcal{G}_B$ into $\{U_{n+1}, U_{n+m}\}$ and $\{U_{n+2}, \dots, U_{n+m-1}\}$. We consider a so called ring structure for group $\mathcal{G}_A$, group $\mathcal{G}_B$ and the merged group $\mathcal{G}' = \mathcal{G}_A \cup \mathcal{G}_B = \{U_1, \dots, U_{n+m}\}$.

The Merge Protocol consists of three rounds during which users U_1 of $\mathcal{G}_A$ and U_{n+1} of $\mathcal{G}_B$ will broadcast keying materials and their respective group's IP address allocation table securely to each other. U_1 and U_{n+1} will also broadcast keying materials that they have computed and received

from each other securely to their own group by encrypting them using their previously established group key, i.e. K_A for group $\mathcal{G}_A$ and K_B for group $\mathcal{G}_B$ respectively. When the Merge Protocol is completed, all users from group $\mathcal{G}_A$ and group $\mathcal{G}_B$ will be merged into the group $\mathcal{G}'$, obtain the new IP address allocation table LIST' and be able to compute the new group key K' . The Merge Protocol is described as follows:

Round 1 :

(i) U_1 first selects a new random $r'_1 \in Z_q^*$ and computes $z'_1 = g^{r'_1} \bmod p$. Next, U_1 signs $U_1 || z'_1 || z_n || \text{addr}[\mathcal{G}_A]$ (where $\text{addr}[\mathcal{G}_A]$ refers to the subnet address used by Group $\mathcal{G}_A$ and $z_n = g^{r_n} \bmod p$ was stored from the previous GKA protocol run) and produces the Rabin's signature (σ'_1, τ'_1) and broadcasts $\{U_1 || z'_1 || z_n || \text{addr}[\mathcal{G}_A] || (\sigma'_1, \tau'_1) || \text{cert}[U_1]\}$.

(ii) U_{n+1} selects new random $r'_{n+1} \in Z_q^*$, computes $z'_{n+1} = g^{r'_{n+1}} \bmod p$ and signs $U_{n+1} || z'_{n+1} || z_{n+m}$ (where $z_{n+m} = g^{r_{n+m}} \bmod p$ was stored from the previous GKA protocol run) and obtains the Rabin's signature $(\sigma'_{n+1}, \tau'_{n+1})$. Next, U_{n+1} broadcasts $\{U_{n+1} || z'_{n+1} || z_{n+m} || (\sigma'_{n+1}, \tau'_{n+1}) || \text{cert}[U_{n+1}]\}$.

Round 2 :

(i) U_1 verifies $\text{cert}[U_{n+1}]$ and signature $(\sigma'_{n+1}, \tau'_{n+1})$ from U_{n+1} . If they are invalid, then U_1 rejects U_{n+1} . Otherwise, U_1 computes DH key $K_{U_1 U_{n+1}} = (z'_{n+1})^{r'_1} = g^{r'_1 r'_{n+1}} \bmod p$ shared with U_{n+1} . U_1 then computes:

$$\begin{aligned} K_A^* &= K_A \cdot (z_2 \cdot y_2^{c_2} \cdot z_n \cdot y_n^{c_n})^{-d_1} \cdot (z_2 \cdot y_2^{c_2} \cdot z_{n+m} \cdot y_{n+m}^{c_{n+m}})^{d'_1} \\ &= g^{d'_1 d_2 + d_2 d_3 + \dots + d_{n-1} d_n + d_{n+m} d'_1} \bmod p. \end{aligned} \quad (5.5)$$

where $d_i = r_i + c_i x_i$ and $c_i = H(z_i)$ for $i = 2, \dots, n, n+m$ and $d'_1 = r'_1 + c'_1 x_1$ where $c'_1 = H(z'_1)$. Next, U_1 encrypts $K_A^* || U_1$ and $K_A^* || \text{LIST_A} || U_1$ using Group $\mathcal{G}_A$'s previously established group key K_A and the DH key $K_{U_1 U_{n+1}}$ respectively and obtains $c_1 = \mathcal{E}_{K_A}(K_A^* || U_1)$ and $c'_1 = \mathcal{E}_{K_{U_1 U_{n+1}}}(K_A^* || \text{LIST_A} || U_1)$ respectively. Finally, U_1 broadcasts $\{U_1 || c_1 || c'_1\}$.

(ii) U_{n+1} first verifies the $\text{cert}[U_1]$ and signature (σ'_1, τ'_1) from U_1 . If they are invalid, then U_{n+1} rejects U_1 . Otherwise, U_{n+1} checks if $\text{addr}[\mathcal{G}_A] = \text{addr}[\mathcal{G}_B]$. If $\text{addr}[\mathcal{G}_A] = \text{addr}[\mathcal{G}_B]$, it means both Group $\mathcal{G}_A$ and $\mathcal{G}_B$ are using the same subnet address and U_{n+1} will prevent address conflicts by changing Group $\mathcal{G}_B$'s subnet address from $\text{addr}[\mathcal{G}_B]$ to $\text{addr}[\mathcal{G}_B]'$. U_{n+1} will also up-

date LIST_B to LIST_B' by updating the subnet address in LIST_B (For example, if $\text{addr}[\mathcal{G}_A] = \text{addr}[\mathcal{G}_B] = 179.156.0.0$, then U_{n+1} will change Group $\mathcal{G}_B$'s subnet address to $\text{addr}[\mathcal{G}_B]' = 179.156.1.0$ and updates LIST_B to LIST_B' using $\text{addr}[\mathcal{G}_B]'$). Next, U_{n+1} computes $K_{U_1 U_{n+1}} = (z'_1)^{r'_{n+1}} = g^{r'_1 r'_{n+1}} \bmod p$, the DH key shared with U_1 . U_{n+1} then computes K_B^* :

$$\begin{aligned} K_B^* &= K_B \cdot (z_n \cdot y_n^{c_n} \cdot z_{n+2} \cdot y_{n+2}^{c_{n+2}})^{d'_{n+1}} \cdot (z_{n+2} \cdot y_{n+2}^{c_{n+2}} \cdot z_{n+m} \cdot y_{n+m}^{c_{n+m}})^{-d_{n+1}} \\ &= g^{d_n d'_{n+1} + d'_{n+1} d_{n+2} + d_{n+2} d_{n+3} + \dots + d_{n+m-1} d_{n+m}} \bmod p \end{aligned} \quad (5.6)$$

where $d_i = r_i + c_i x_i$ and $c_i = H(z_i)$ for $n \leq i \neq n+1 \leq n+m$ and $d'_{n+1} = r'_{n+1} + c'_{n+1} x_{n+1}$ where $c'_{n+1} = H(z'_{n+1})$. Next, U_{n+1} encrypts $K_B^* || \text{addr}[\mathcal{G}_B]' || U_{n+1}$ and $K_B^* || \text{LIST_B}' || U_{n+1}$ using Group $\mathcal{G}_B$'s previously established group key K_B and the DH key $K_{U_1 U_{n+1}}$ respectively and obtains $c_{n+1} = \mathcal{E}_{K_B}(K_B^* || \text{addr}[\mathcal{G}_B]' || U_{n+1})$ and $c'_{n+1} = \mathcal{E}_{K_{U_1 U_{n+1}}}(K_B^* || \text{LIST_B}' || U_{n+1})$ respectively where $\text{addr}[\mathcal{G}_B]' = \text{addr}[\mathcal{G}_B]$ and $\text{LIST_B}' = \text{LIST_B}$ if there are no address conflicts. Finally, U_{n+1} broadcasts $\{U_{n+1} || c_{n+1} || c'_{n+1}\}$.

Round 3 :

(i) U_1 decrypts c'_{n+1} and obtains K_B^* , LIST_B' and the identity U_{n+1} . If the identity U_{n+1} is not decrypted correctly, then U_1 rejects c'_{n+1} . Otherwise, U_1 combines LIST_A with LIST_B' to form the merged group's IP allocation table LIST'. U_1 then encrypts $K_B^* || \text{LIST}' || U_1$ using Group $\mathcal{G}_A$'s previously established group key K_A and obtains $c''_1 = \mathcal{E}_{K_A}(K_B^* || \text{LIST}' || U_1)$ and broadcasts $\{U_1 || c''_1\}$ to group $\mathcal{G}_A$.

(ii) U_{n+1} decrypts c''_1 and obtains K_A^* , LIST_A and the identity U_1 . If the identity U_1 is not decrypted correctly, then U_{n+1} rejects c''_1 . Otherwise, U_{n+1} combines LIST_B' with LIST_A to form the merged IP allocation table LIST'. U_{n+1} then encrypts $K_A^* || \text{LIST}' || U_{n+1}$ using Group $\mathcal{G}_B$'s previously established group key K_B and obtains $c''_{n+1} = \mathcal{E}_{K_B}(K_A^* || \text{LIST}' || U_{n+1})$ and broadcasts $\{U_{n+1} || c''_{n+1}\}$ to group $\mathcal{G}_B$.

Key Computation :

(i) Group $\mathcal{G}_A$ users U_i for $2 \leq i \leq n$ first decrypt c_1 using Group $\mathcal{G}_A$'s previously established group key K_A and obtain K_A^* and the identity U_1 . If the identity U_1 is not decrypted correctly, then each U_i rejects c_1 . Otherwise, U_i accepts K_A^* . Next, U_i decrypts c''_1 using Group $\mathcal{G}_A$'s previously established

group key K_A and obtain K_B^* , LIST' and the identity U_1 . If the identity U_1 is not decrypted correctly, then each U_i rejects c_1'' . Otherwise, each U_i stores LIST' as the new IP allocation table and accepts K_B^* .

(ii) Group $\mathcal{G}_B$ users U_j for $n+2 \leq j \leq n+m$ first decrypt c_{n+1} and obtain K_B^* , $\text{addr}[\mathcal{G}_B]'$ and identity U_{n+1} . If the identity U_{n+1} is not decrypted correctly, each U_j rejects c_{n+1} . Otherwise, each U_j stores K_B^* for key computation later. U_j also checks $\text{addr}[\mathcal{G}_B]'$ for address conflict. If there is an address conflict, then $\text{addr}[\mathcal{G}_B]' \neq \text{addr}[\mathcal{G}_B]$, where $\text{addr}[\mathcal{G}_B]$ is the current Group $\mathcal{G}_B$'s subnet address. Then each U_j will change its IP address according to $\text{addr}[\mathcal{G}_B]'$ and stores K_B^* . Next, U_j decrypt c_{n+1}'' and obtain K_A^* , LIST' and the identity U_{n+1} respectively. If the identity U_{n+1} is not decrypted correctly, then each U_j rejects c_{n+1}'' . Otherwise, each U_j stores LIST' as the new IP allocation table and stores K_A^* for key computation later.

(iii) Finally, all users U_i for $1 \leq i \leq n+m$ in the merged group $\mathcal{G}'$ compute the new group key K' as follows:

$$\begin{aligned} K' &= K_A^* \cdot K_B^* \\ &= g^{d_1' d_2 + d_2 d_3 + \dots + d_n d_{n+1}' + d_{n+1}' d_{n+2} + d_{n+2} d_{n+3} + \dots + d_{n+m} d_1'} \mod p. \end{aligned} \quad (5.7)$$

where $d_i = r_i + c_i x_i$ for $2 \leq i \leq n+m$, $d_1' = r_1' + c_1' x_1$ and $d_{n+1}' = r_{n+1}' + c_{n+1}' x_{n+1}$.

5.5.4 Leave Protocol

A leaving user U_l will send a leave notification to announce its departure from the group. The group key will be changed to protect future MANET communications from the leaving user. Let $\mathcal{G} = \{U_1, \dots, U_n\}$ denote the original group and $\mathcal{G}' = \mathcal{G} \setminus \{U_l\}$ denotes the new group. We also consider a so called ring structure for group $\mathcal{G}$ and group $\mathcal{G}'$.

The Leave Protocol comprises of two rounds. In the first round, all users of $\mathcal{G}'$ (i.e. all remaining users) will update their IP address allocation table by removing the IP address of U_l . In addition, all *odd-indexed* users of $\mathcal{G}'$ compute and broadcast their new keying materials. In Round 2, all users of $\mathcal{G}'$ compute and broadcast new keying materials. Finally, all users of $\mathcal{G}'$ will compute the new group key K' . The Leave Protocol is described as follows:

Round 1 :

(i) All remaining users $U_i \in \mathcal{G}'$ for $1 \leq i \neq l \leq n$ update **LIST** to **LIST'** by deleting the IP address of the leaving user U_l .

(ii) Besides updating their allocation table, all remaining *odd-indexed* users $U_j \neq U_l$ for $1 \leq j \neq l \leq n$ and j is odd select new random $r'_j \in Z_q^*$, compute $z'_j = g^{r'_j} \bmod p$ and broadcast $\{U_j || z'_j\}$.

Round 2 : All remaining users $U_i \in \mathcal{G}'$ for $1 \leq i \neq l \leq n$ compute $\hat{X}'_i$ as follows:

$$\hat{X}'_i = \begin{cases} \left(\frac{z'_{i+1} \cdot y'_{i+1}}{z'_{i-1} \cdot y'_{i-1}} \right)^{d'_i} & \text{if } i \neq l, l-1, l+1, \\ \left(\frac{z'_{l+1} \cdot y'_{l+1}}{z'_{l-2} \cdot y'_{l-2}} \right)^{d'_{l-1}} & \text{if } i = l-1, \\ \left(\frac{z'_{l+2} \cdot y'_{l+2}}{z'_{l-1} \cdot y'_{l-1}} \right)^{d'_{l+1}} & \text{if } i = l+1. \end{cases} \quad (5.8)$$

for $d'_i = r'_i + c'_i x_i$ and $c'_i = H(z'_i)$ where $z'_i = z_i$, $d'_i = d_i$, $r'_i = r_i$ and $c'_i = c_i$ for $i \neq l$ and i is even. Finally, each remaining user $U_i \in \mathcal{G}'$ for $1 \leq i \neq l \leq n$ computes $\mathcal{Z}' = \prod_{j=1, j \neq l}^n z'_j \bmod p$ and signs $U_i || z'_i || \hat{X}'_i || \mathcal{Z}'$ to obtain Rabin's signature (σ'_i, τ'_i) and broadcasts $\{U_i || \hat{X}'_i || (\sigma'_i, \tau'_i)\}$. All remaining users store $\mathcal{Z}'$ for verification in the next round. We assume U_1 to be a trusted controller such that U_1 is the last user to broadcast its message after all other users have broadcast their messages.

Authentication and Key Computation: Each $U_i \in \mathcal{G}'$ for $1 \leq i \neq l \leq n$ verifies the Rabin's signature (σ'_j, τ'_j) of message $U_j || z'_j || \hat{X}'_j || \mathcal{Z}'$ for $1 \leq j \neq i \leq n$ and $j \neq l$, where $\mathcal{Z}'$ was stored in Round 2. If the signature verifications are correct, then from equation (5.8) and using similar calculations as mentioned in the Authentication and Key Computation Round of the proposed IKA protocol in Section 5.5.1, each $U_i \in \mathcal{G}'$ for $1 \leq i \neq l \leq n$ obtains $g^{d'_j d'_{j+1}} \bmod p$ for $1 \leq j \neq l \leq n$ where $d'_j = r'_j + c'_j x_j$ and $d'_j = d_j$ if $j \neq l$ is even and $d'_{n+1} = d'_1$. Then each $U_i \in \mathcal{G}'$ for $1 \leq i \neq l \leq n$ computes the new group key K' as follows:

$$\begin{aligned} K' &= \prod_{j=1, j \neq l}^n g^{d'_j d'_{j+1}} \\ &= g^{d'_1 d'_2 + \dots + d'_{l-2} d'_{l-1} + d'_{l-1} d'_{l+1} + \dots + d'_n d'_1} \bmod p \end{aligned} \quad (5.9)$$

where $d'_j = r'_j + c'_j x_j$ for $1 \leq j \neq l \leq n$ and $d'_j = d_j$ if $j \neq l$ is even and $d'_{n+1} = d'_1$. If a user $U_i \in \mathcal{G}'$

for $1 \leq i \neq l \leq n$ detects that the signature (σ'_a, τ'_a) is incorrect, then all users will exclude U_a from the GKA by using the new $\hat{X}'_{a-1}$ and $\hat{X}'_{a+1}$ as mentioned in the Initial Key Agreement Protocol.

5.6 Security Analysis

In this section, we provide security analysis for our proposed protocols, namely the IKA, Join, Merge and Leave protocols. We show that our protocols are secure as follows:

5.6.1 Security Analysis of IKA Protocol

We prove that an adversary $\mathcal{A}$ is not able to obtain the group key K from the proposed IKA protocol and the proposed IKA protocol is secure against insider impersonation.

Theorem 8 *An adversary $\mathcal{A}$ is unable to obtain the group key K in the proposed IKA protocol provided the CDH problem is difficult and the Rabin signature scheme and the VBD protocol are secure.*

Proof: Suppose an adversary $\mathcal{A}$ tries to impersonate a group member U_i (for some i) to obtain a group key, $\mathcal{A}$ first generates random $z_i = g^{r_i} \bmod p$ in Round 1, then in Round 2, $\mathcal{A}$ has to compute a valid $\hat{X}_i = g^{(r_i+c_i x_i)(r_{i+1}+c_{i+1} x_{i+1})-(r_i+c_i x_i)(r_{i-1}+c_{i-1} x_{i-1})}$. As $\mathcal{A}$ does not know the private key x_i of U_i and the CDH problem is assumed to be difficult, therefore $\mathcal{A}$ is not able to produce a valid $\hat{X}_i$. Furthermore, $\mathcal{A}$ needs to compute Rabin signature (σ_i, τ_i) of the message $U_i || z_i || \hat{X}_i || \prod_{j=1}^n z_j || \text{addr}[U_i]$, where σ_i is the modular square root of $\sigma_i^2 = a_i \bmod \bar{N}_i$ where $a_i = H(U_i || z_i || \hat{X}_i || \prod_{j=1}^n z_j || \text{addr}[U_i] || \tau_i)$, and τ_i is a random chosen by U_i such that $a_i^{\frac{\bar{p}_i-1}{2}} = 1 \bmod \bar{p}_i$ and $a_i^{\frac{\bar{q}_i-1}{2}} = 1 \bmod \bar{q}_i$ where $(\bar{p}_i, \bar{q}_i)$ is the private key of U_i . The validity of Rabin signature (σ_i, τ_i) is checked by the other group members in the Authentication and Key Computation Round by checking the equation $\sigma_i^2 = a_i \bmod \bar{N}_i$. If the equation $\sigma_i^2 = a_i \bmod \bar{N}_i$ is not correct, then the other group members will know that the Rabin signature (σ_i, τ_i) is not correct and U_i is not a valid user. As $\mathcal{A}$ does not know the private key $(\bar{p}_i, \bar{q}_i)$ of U_i and the Rabin signature scheme is assumed to be secure, therefore, $\mathcal{A}$ is not able to produce a valid Rabin signature (σ_i, τ_i) . Hence, the adversary $\mathcal{A}$ is not able to impersonate the user U_i to establish a group key. Now, consider that an adversary $\mathcal{A}$ can only obtain the public information sent by the group members. Then, the

proposed IKA protocol (without the signature $((\sigma_j, \tau_j)$ for $1 \leq j \leq n$) in round 2 and $\text{addr}[\mathbf{U}_i]$ in round 1 and round 2) is just a VBD protocol (that is described in Section 4.4.1 of Chapter 4). By Theorem 5, it is shown that the VBD protocol is secure. Therefore, the adversary $\mathcal{A}$ is not able to obtain the group key established by a group. ■

Theorem 9 *The proposed IKA protocol is secure against insider impersonation if the Rabin signature scheme is secure.*

Proof: Assume that there are k ($< n - 1$) group members in a group of n members impersonating a honest user $\mathbf{U}_i$ (for some i) to establish a new group key, then there is at least one group member, say $\mathbf{U}_j$ (who is not one of the k impersonators) sending a new z'_j in round 1 which is different from that of the previously sent. Therefore, $\mathcal{Z}' = \prod_{k=1}^n z'_k$ will be different from the previous $\mathcal{Z}$ sent. In this case, the k number of impersonators need to produce a valid Rabin signature (σ'_i, τ'_i) of the message $\mathbf{U}_i || z_i || \hat{X}_i || \prod_{k=1}^n z'_k || \text{addr}[\mathbf{U}_i]$ for the user $\mathbf{U}_i$ in round 2. Although the k ($< n - 1$) impersonators can produce a valid $\hat{X}_i$ without knowing the private key x_i of $\mathbf{U}_i$, however, it is impossible for the k number of impersonators to produce a valid Rabin signature (σ'_i, τ'_i) of the message $\mathbf{U}_i || z_i || \hat{X}_i || \prod_{k=1}^n z'_k || \text{addr}[\mathbf{U}_i]$ as they do not know the private key $(\bar{p}_i, \bar{q}_i)$ of $\mathbf{U}_i$ and the Rabin signature scheme is assumed to be secure. Therefore, it is not possible for any k ($< n - 1$) insiders to impersonate any user in the group to establish a new group key. ■

5.6.2 Security Analysis of Join Protocol

In this section, we first show that the Join Protocol achieves backward secrecy, that is, the new user $\mathbf{U}_{n+1}$ cannot obtain the previously established group key K from new group key K' with the information known by the new user $\mathbf{U}_{n+1}$. Next, we show that an adversary $\mathcal{A}$ cannot obtain the new group key K' from the proposed Join protocol.

Theorem 10 *The proposed Join protocol achieves backward secrecy in the information-theoretic sense.*

Proof: Suppose a new user $\mathbf{U}_{n+1}$ tries to obtain the previous group key $K = g^{d_1 d_2 + \dots + d_{n-1} d_n + d_n d_1}$ where $d_i = r_i + c_i x_i$ and $c_i = H(z_i)$ for $1 \leq i \leq n$ from the Join Protocol, then $\mathbf{U}_{n+1}$ first obtained $K^* = g^{d'_1 d_2 + d_2 d_3 + \dots + d_{n-1} d_n + d_n d_{n+1} d'_1}$ where $d_i = r_i + c_i x_i$ and $c_i = H(z_i)$ for $2 \leq i \leq n+1$, $d'_1 = r'_1 + c'_1 x_1$

and $c'_1 = H(g^{r'_1})$ and a new group key $K' = g^{d'_1 d_2 + d_2 d_3 + \dots + d_{n-1} d_n + d_n d_{n+1} + d_{n+1} d'_1}$ from the Join protocol. The difference between K and K^* are the terms $g^{(r_1+c_1x_1)(r_2+c_2x_2)}$ and $g^{(r'_1+c'_1x_1)(r_2+c_2x_2)}$; and $g^{(r_n+c_nx_n)(r_1+c_1x_1)}$ and $g^{(r_{n+1}+c_{n+1}x_{n+1})(r'_1+c'_1x_1)}$ respectively. As r'_1 and r_1 are random and independent to each other, hence $g^{r'_1}$ and $z_1 = g^{r_1}$ are independent to each other. Therefore $c'_1 = H(g^{r'_1})$ and $c_1 = H(z_1)$ are independent to each other as the output of the hash function H is independent. As a result, the exponent $r'_1 + c'_1x_1$ and $r_1 + c_1x_1$ are independent with each other, therefore, $g^{(r'_1+c'_1x_1)(r_2+c_2x_2)}$ and $g^{(r_1+c_1x_1)(r_2+c_2x_2)}$ are independent to each other. Similarly, the terms $g^{(r_{n+1}+c_{n+1}x_{n+1})(r'_1+c'_1x_1)}$ and $g^{(r_n+c_nx_n)(r_1+c_1x_1)}$ are independent to each other. Therefore, K and K^* will look random and independent to each other. Furthermore, there is no information about r'_1 being send out except in K^* and K' is obtained by multiplying K^* with $g^{(r_n+c_nx_n)(r_{n+1}+c_{n+1}x_{n+1})}$. Hence, the new user will not be able to derive K from K^* and K' . Therefore, we conclude that the Join protocol achieves backward secrecy in the information-theoretic sense. ■

Next, we show that the outsider malicious adversary $\mathcal{A}$ is unable to obtain the new group key K' from the Join Protocol.

Theorem 11 *An adversary $\mathcal{A}$ is not able to obtain the new group key K' from the proposed Join Protocol provided the Rabin signature scheme and the symmetric key encryption scheme SKE are secure and the CDH problem is difficult.*

Proof: In this proof, we consider two cases, that is, (a) an adversary $\mathcal{A}$ tries to impersonate a user U_i (for some $i = n$ or $i = n+1$) to establish a new group key, (b) an adversary $\mathcal{A}$ tries to obtain a new group key from the public exchange information. For case (a), suppose $\mathcal{A}$ wants to impersonate U_n (U_{n+1}), then in Round 1, $\mathcal{A}$ computes $z_n = g^{r_n}$ ($z_{n+1} = g^{r_{n+1}}$). Next, $\mathcal{A}$ has to compute the Rabin signature (σ'_n, τ'_n) of the message $U_n || z_n || \text{addr}[U_{n+1}]$ (the Rabin signature $(\sigma_{n+1}, \tau_{n+1})$ of the message $U_{n+1} || z_{n+1}$). The validity of Rabin signature (σ'_n, τ'_n) is checked by U_{n+1} in the Key Computation Round (the validity of Rabin signature $(\sigma_{n+1}, \tau_{n+1})$ is checked by U_1 and U_n in Round 2 and 3 respectively). If the Rabin signature (σ'_n, τ'_n) is not correct, then U_{n+1} rejects U_n (if the Rabin signature $(\sigma_{n+1}, \tau_{n+1})$ is not correct, then U_1 and U_n rejects U_{n+1}). As $\mathcal{A}$ does not know the private key $(\bar{p}_i, \bar{q}_i)$ of U_i and the Rabin signature scheme is assumed to be secure, hence $\mathcal{A}$ is not able to produce the Rabin signature (σ'_n, τ'_n) ($(\sigma_{n+1}, \tau_{n+1})$). Furthermore, $\mathcal{A}$ does not know the

private key x_i of U_i . Hence $\mathcal{A}$ is not able to impersonate user U_i . Therefore, it is not possible for an adversary $\mathcal{A}$ to obtain the new group key in this case. Now, we consider the case (b) as follows: As an adversary $\mathcal{A}$ is not a joining new user U_{n+1} , $\mathcal{A}$ can only obtain the public exchange information, that is, $z_n, y_n, z_{n+1}, y_{n+1}, c_1, c_n$ and c'_n . As the new group key $K' = K^* \cdot g^{(r_n+c_nx_n)(r_{n+1}+c_{n+1}x_{n+1})}$ where K^* and $g^{(r_n+c_nx_n)(r_{n+1}+c_{n+1}x_{n+1})}$ are the plaintext of c_1 and c_n respectively, therefore, it is not possible for $\mathcal{A}$ to recover K^* and $g^{(r_n+c_nx_n)(r_{n+1}+c_{n+1}x_{n+1})}$ from c_1 and c_n respectively as it is assumed that the symmetric key encryption scheme SKE is secure. On the other hand, if the adversary $\mathcal{A}$ can obtain $g^{(r_n+c_nx_n)(r_{n+1}+c_{n+1}x_{n+1})}$ from z_n, y_n, z_{n+1} and y_{n+1} , then the adversary $\mathcal{A}$ can obtain K^* from c'_n and $\mathcal{A}$ obtains the new group key $K' = K^* \cdot g^{(r_n+c_nx_n)(r_{n+1}+c_{n+1}x_{n+1})}$. But this is not possible for the adversary $\mathcal{A}$ to obtain $g^{(r_n+c_nx_n)(r_{n+1}+c_{n+1}x_{n+1})}$ as this is the CDH problem and is assumed to be difficult. Therefore, the adversary $\mathcal{A}$ is not possible to obtain the new group key K' . ■

5.6.3 Security Analysis of Merge Protocol

In this section, we first show that the Merge Protocol provides backward secrecy, that is, users in Group $\mathcal{G}_B$ cannot obtain Group $\mathcal{G}_A$'s previously established group key K_A and users in Group $\mathcal{G}_A$ cannot obtain Group $\mathcal{G}_B$'s previously established group key K_B .

Theorem 12 *The proposed Merge protocol achieves backward secrecy provided the CDH problem is difficult.*

Proof: In this theorem, we need to show that any user in group $\mathcal{G}_B$ ($\mathcal{G}_A$) is not able to obtain the previous group key K_A (K_B) of $\mathcal{G}_A$ ($\mathcal{G}_B$). Suppose a user U_k ($n+1 \leq k \leq n+m$) in group $\mathcal{G}_B$ tries to obtain the previous group key $K_A = g^{d_1d_2+d_2d_3+\dots+d_{n-1}d_n+d_nd_1}$ where $d_i = r_i + c_ix_i$ and $c_i = H(z_i)$ for $1 \leq i \leq n$ of group $\mathcal{G}_A$ from the Merge protocol, then U_{n+1} first obtained z'_1 and $K_A^* = g^{d'_1d_2+d_2d_3+\dots+d_{n-1}d_n+d_{n+m}d'_1}$ where $d_i = r_i + c_ix_i$ and $c_i = H(z_i)$ for $i = 2, 3, \dots, n, n+m$, $d'_1 = r'_1 + c'_1x_1$ and $c'_1 = H(z'_1)$ from the Merge protocol. The difference between K_A and K_A^* are the terms $g^{(r_1+c_1x_1)(r_2+c_2x_2)}$ and $g^{(r'_1+c'_1x_1)(r_2+c_2x_2)}$; and $g^{(r_n+c_nx_n)(r_1+c_1x_1)}$ and $g^{(r_{n+m}+c_{n+m}x_{n+m})(r'_1+c'_1x_1)}$ respectively. As r'_1 and r_1 are random and independent to each other, hence $z'_1 = g^{r'_1}$ and $z_1 = g^{r_1}$ are independent to each other. Therefore $c'_1 = H(z'_1)$ and $c_1 = H(z_1)$ are independent to each other as the output of the hash function H is independent. As a result,

the exponent $r'_1 + c'_1 x_1$ and $r_1 + c_1 x_1$ are independent with each other, therefore, $g^{(r'_1 + c'_1 x_1)(r_2 + c_2 x_2)}$ and $g^{(r_1 + c_1 x_1)(r_2 + c_2 x_2)}$ are independent to each other. Similarly, $g^{(r_{n+m} + c_{n+m} x_{n+m})(r'_1 + c'_1 x_1)}$ and $g^{(r_n + c_n x_n)(r_1 + c_1 x_1)}$ are independent to each other. Therefore, K_A and K_A^* will look random and independent to each other. Furthermore, if U_k can obtain $g^{(r_n + c_n x_n)(r_1 + c_1 x_1)}$, $g^{(r_1 + c_1 x_1)(r_2 + c_2 x_2)}$, $g^{(r_{n+m} + c_{n+m} x_{n+m})(r'_1 + c'_1 x_1)}$ (except user U_{n+m} which can compute this as he knows r_{n+m} , c_{n+m} and x_{n+m}) and $g^{(r'_1 + c'_1 x_1)(r_2 + c_2 x_2)}$ from z_n , y_n , z_1 , y_1 , z'_1 , z_{n+m} , y_{n+m} , z_2 and y_2 . Then, U_k can solve the CDH problem, but this contradicts to the assumption that the CDH problem is difficult. Hence, the user U_k in group $\mathcal{G}_B$ will not be able to derive K_A from K_A^* . The similar proof can be shown that any user in group $\mathcal{G}_A$ is not able to obtain the previous group key K_B of $\mathcal{G}_B$. Therefore, we conclude that the Merge protocol achieves backward secrecy if the CDH problem is difficult. ■

Next, we show that an adversary $\mathcal{A}$ is not able to obtain the new group key K' from the Merge Protocol.

Theorem 13 *An adversary $\mathcal{A}$ is not able to obtain the new group key K' from the proposed Merge Protocol provided the Rabin signature scheme and the symmetric key encryption scheme SKE are secure and the CDH problem is difficult.*

Proof: In this proof, we consider two cases, that is, (a) an adversary $\mathcal{A}$ tries to impersonate a user U_i (for some $i = 1$ or $i = n+1$) to establish a new group key, (b) an adversary $\mathcal{A}$ tries to obtain a new group key from the public exchange information. For case (a), suppose $\mathcal{A}$ wants to impersonate U_1 (U_{n+1}), then in Round 1, $\mathcal{A}$ computes $z'_1 = g^{r'_1}$ ($z'_{n+1} = g^{r'_{n+1}}$). Next, $\mathcal{A}$ has to compute the Rabin signature (σ'_1, τ'_1) of the message $U_1 || z'_1 || z_n || \text{addr}[\mathcal{G}_A]$ (the Rabin signature $(\sigma'_{n+1}, \tau'_{n+1})$ of the message $U_{n+1} || z'_{n+1} || z_{n+m}$). The validity of Rabin signature (σ'_1, τ'_1) ($(\sigma'_{n+1}, \tau'_{n+1})$) is checked by U_{n+1} (U_1) in Round 2. If the Rabin signature (σ'_1, τ'_1) is not correct, then U_{n+1} rejects U_1 (if the Rabin signature $(\sigma'_{n+1}, \tau'_{n+1})$ is not correct, then U_1 rejects U_{n+1}). As $\mathcal{A}$ does not know the private key $(\bar{p}_i, \bar{q}_i)$ of U_i and the Rabin signature scheme is assumed to be secure, hence $\mathcal{A}$ is not able to produce the Rabin signature (σ'_1, τ'_1) ($(\sigma'_{n+1}, \tau'_{n+1})$). Furthermore, $\mathcal{A}$ does not know the private key x_i of U_i . Hence $\mathcal{A}$ is not able to impersonate user U_i . Therefore, it is not possible for an adversary to obtain the new group key in this case. Now, we consider the case (b) as follows: As an adversary $\mathcal{A}$ is neither a legitimate user in group $\mathcal{G}_A$ nor group $\mathcal{G}_B$, $\mathcal{A}$ can only obtain the public exchange

information, that is, $z'_1, z_n, z'_{n+1}, z_{n+m}, c_1, c'_1, c''_1, c_{n+1}, c'_{n+1}$ and c''_{n+1} . As the new group key $K' = K_A^* \cdot K_B^*$ where K_A^* is the plaintext of c_1 and c''_{n+1} and K_B^* is the plaintext of c_{n+1} and c'_1 respectively, therefore, it is not possible for $\mathcal{A}$ to recover K_A^* from c_1 and c''_{n+1} and K_B^* from c_{n+1} and c'_1 respectively as it is assumed that the symmetric key encryption scheme SKE is secure. On the other hand, if the adversary $\mathcal{A}$ can obtain $g^{r'_1 r'_{n+1}}$ from z'_1 and z'_{n+1} , then the adversary $\mathcal{A}$ can obtain K_A^* and K_B^* from c'_1 and c'_{n+1} respectively and $\mathcal{A}$ obtains the new group key $K' = K_A^* \cdot K_B^*$. But this is not possible for the adversary $\mathcal{A}$ to obtain $g^{r'_1 r'_{n+1}}$ as this is the CDH problem and is assumed to be difficult. Therefore, the adversary $\mathcal{A}$ is not possible to obtain new group key K' . ■

5.6.4 Security Analysis of Leave Protocol

In this section, we first show that the proposed Leave protocol achieves forward secrecy, that is, the leaving user U_l cannot obtain the new group key K' from previous group key K with the information known by the leaving user U_l . Next, we show that an adversary $\mathcal{A}$ cannot obtain the new group key K' from the proposed Leave protocol.

Theorem 14 *The proposed Leave protocol achieves forward secrecy provided the CDH problem is difficult and VBD protocol is secure.*

Proof: As the leaving user U_l knows the previous group key $K = g^{d_n d_1} \cdot \prod_{i=1}^{n-1} g^{d_i d_{i+1}}$ (which is of n terms of the form g^{de}) where $d_i = r_i + c_i x_i$ and $c_i = H(z_i)$ for $1 \leq i \leq n$ and all the terms $g^{d_i d_{i+1}}$ for $1 \leq i \leq n-1$ and $g^{d_n d_1}$, but the new group key is of the form $K' = g^{d'_1 d'_2 + d'_2 d'_3 + \dots + d'_{l-2} d'_{l-1} + d'_{l-1} d'_{l+1} + \dots + d'_n d'_1}$ (which is of $n-1$ terms of the form g^{de}) where $d'_i = r'_i + c'_i x_i$ and $c'_i = H(z'_i)$ for $1 \leq i \neq l \leq n$, $r'_j = r_j$ and $c'_j = c_j$ for $1 \leq j \neq l \leq n$ and j is even. Furthermore, each r'_j is independent of r_j for $1 \leq j \neq l \leq n$ and j is odd, hence $z'_j = g^{r'_j}$ and $z_j = g^{r_j}$ are independent to each other. Therefore, $c'_j = H(z'_j)$ and $c_j = H(z_j)$ are independent to each other as the output of the hash function H is independent. As a result, the exponent $r'_j + c'_j x_j$ and $r_j + c_j x_j$ are independent with each other, therefore, $g^{(r'_j + c'_j x_j)(r_{j+1} + c_{j+1} x_{j+1})}$ is independent of $g^{(r_j + c_j x_j)(r_{j+1} + c_{j+1} x_{j+1})}$. On the other hand, if the leaving user U_l can compute $g^{(r'_j + c'_j x_j)(r_{j+1} + c_{j+1} x_{j+1})}$ from $g^{r'_j + c'_j x_j}$ and $g^{r_{j+1} + c_{j+1} x_{j+1}}$ which is the CDH problem, then this contradicts to the fact that the CDH problem is difficult. Furthermore, the two terms $g^{(r'_{l-2} + c'_{l-2} x_{l-2})(r'_{l-1} + c'_{l-1} x_{l-1})}$ and $g^{(r'_{l-1} + c'_{l-1} x_{l-1})(r'_{l+1} + c'_{l+1} x_{l+1})}$ only appeared in K' , but not

in K . Therefore, K' is independent of K . Hence, even though the leaving user U_l knows all $g^{d_i d_{i+1}}$ for $1 \leq i \leq n-1$ and $g^{d_n d_1}$ where $d_i = r_i + c_i x_i$ and $c_i = H(z_i)$ for $1 \leq i \leq n$, it is not possible for U_l to obtain a new group key K' from K . This leads to the case that the leaving user U_l can only derive the new group key K' through the public exchange information, that is, round 1 and round 2 information sent by the remaining group members. But the Leave protocol (without the signature information sent) is just the VBD protocol, by Theorem 5, it is shown that the VBD protocol is secure. Therefore, the leaving user U_l is not able to obtain the new group key K' . Therefore, the leave protocol achieves forward secrecy. ■

Theorem 15 *An adversary $\mathcal{A}$ (not a leaving user U_l) cannot obtain the new group key K' in the proposed Leave protocol provided that the CDH problem is difficult and the Rabin signature scheme and the VBD protocol are secure.*

Proof: In this proof, we consider two cases, that is, (a) an adversary $\mathcal{A}$ tries to impersonate a user U_i (for some i is odd such that $i \neq l$) to establish a new group key, (b) an adversary $\mathcal{A}$ tries to obtain a new group key from the public exchange information. For case (a), it is already shown in Theorem 8 that no adversary can impersonate any user in the group as they do not know the private key x_i of U_i to compute a valid $\hat{X}'_i$ provided that the CDH problem is difficult. Furthermore, $\mathcal{A}$ cannot compute a valid signature (σ'_i, τ'_i) of the message $U_i || z'_i || \hat{X}'_i || \prod_{j=1, j \neq l}^n z'_j$ sent in Round 2 as $\mathcal{A}$ does not know the private key $(\bar{p}_i, \bar{q}_i)$ of U_i and the Rabin signature scheme is secure. Therefore, it is not possible for an adversary to obtain the new group key in this case. Now, we consider the case (b) as follows: As an adversary $\mathcal{A}$ is not a leaving user U_l , $\mathcal{A}$ can only obtain the public exchange information, that is, $z_i, \hat{X}_i$ for $1 \leq i \leq n$ and $z'_j, \hat{X}'_j$ for $1 \leq j \neq l \leq n$ from IKA protocol and the Leave protocol respectively. It is noted that z'_i and z_i are independent to each other for i is odd and $i \neq l$, hence $c'_i = H(z'_i)$ and $c_i = H(z_i)$ are independent to each other for i is odd and $i \neq l$ as the output of the hash function H is independent, and $\hat{X}'_i$ and $\hat{X}_i$ are independent to each other for $1 \leq i \leq n$ such that $i \neq l, l-1, l+1$. Furthermore, the number of terms of the form g^{de} in K and K' are different by 1. Therefore, the key K and K' is independent to each other. Hence, the adversary $\mathcal{A}$ can only derive the key K' from the public exchange information in the Leave protocol. But the Leave protocol (without the signature information sent) is just the VBD protocol, by Theorem 5, it is shown that the VBD protocol is secure. Therefore, the adversary $\mathcal{A}$

is not able to obtain the new group key K' from the public exchange information. ■

5.7 Complexity Analysis

In Table 5.1, we compare our proposed protocols with four other GKA protocols in terms of communication and computation costs for dynamic events. We focus on the number of rounds, the total number of messages, the number of exponentiations, the number of signature generations, and the number of signature verifications. The four protocols, GDH [STW00], TGDH [KPT04b], STR [KPT01] and BD [BD95], are based on [AKNR⁺04] and [KPT04b]. The details of the four GKA protocols are provided in Section 2.

Table 5.1: Computation and Communication Costs

Protocol		Communication Costs		Computation Costs		
		Round	Messages	Exp.	Signature Generation	Signature Verification
GDH [STW00]	Join	4	$n + 3$	$n + 3$	4	$n + 3$
	Leave	1	1	$n - 1$	1	1
	Merge	$m + 3$	$n + 2m + 1$	$n + 2m + 1$	$m + 3$	$n + 2m + 1$
TGDH [KPT04b]	Join	2	3	$3h - 3$	2	3
	Leave	1	1	$3h - 3$	1	1
	Merge	$\lceil \log_2 k \rceil + 1$	$2k$	$3h - 3$	$\lceil \log_2 k \rceil + 1$	$\lceil \log_2 k \rceil$
STR [KPT01]	Join	2	3	4	2	3
	Leave	1	1	$\frac{3n}{2} + 2$	1	1
	Merge	2	$k + 1$	$3m + 1$	2	3
BD [BD95]	Join	2	$2n + 2$	3^{*1}	2	$n + 3$
	Leave	2	$2n - 2$	3^{*1}	2	$n + 1$
	Merge	2	$2n + 2m$	3^{*1}	2	$n + m + 2$
Prop. Sch.	Join	3	5	3^{*2}	1	1
	Leave	2	$v + n - 2$	4^{*3}	1	$n - 1$
	Merge	3	$6(k - 1)$	5^{*4}	1	1

*1 : For BD protocol, all users perform 3 exponentiations.

*2 : In our Join Protocol, only U_1 and U_{n+1} perform 3 exponentiations each.

*3 : For our Leave Protocol, users U_i for i is odd and U_j for j is even perform 4 and 3 exponentiations respectively.

*4 : In our Merge Protocol, only U_1 and U_{n+1} performs 5 exponentiations.

Our proposed protocols include the private keys x_i of all users U_i and provide secure IP address autoconfiguration and key authentication. As studies [CKM00,Mir] have shown that symmetric key

cryptography energy requirements are several orders of magnitude lower than modular exponentiations, we assume that the computational complexity of symmetric key encryption/decryption to be negligible as compared to exponentiations. The current group size, merging users, merging groups, remaining *odd-indexed* users and height of key tree are denoted as n , m , k , v and h respectively.

Join Protocol

Despite providing autoconfiguration, our protocol only requires three rounds and five messages whereas GDH requires four rounds and $\mathcal{O}(n)$ messages and BD is the most expensive, requiring $2n + 2$ messages. In terms of exponentiations, our protocol is the most efficient as only users U_1 , U_n and U_{n+1} perform three, two and three exponentiations respectively while the rest need not perform any exponentiations. The BD, however, requires *each* user to perform *three* exponentiations. Our protocol is the most efficient in terms of signature generation and verification. Only U_n and U_{n+1} needs to generate and verify *one* signature each while U_1 verifies *one* signature and all other users need not generate or verify any signatures.

Leave Protocol

Our protocol is more efficient than BD as our protocol requires only users $U_j \neq U_l$ for j is odd to broadcast in the first round while BD requires $2n - 2$ messages and is the most expensive protocol. For computational costs, our protocol requires only users $U_j \neq U_l$ for j is odd to perform *four* exponentiations while $U_k \neq U_l$ for k is even to perform *three* exponentiations as compared to BD which requires each user to perform *three* exponentiations and do not provide key authentication. GDH and STR are also less efficient as they scale linearly with the group size n and do not provide key authentication. In terms of signature generation, our protocol requires only *one* signature generation. BD requires the most number $(n + 1)$ of signature verifications while our proposed protocol requires $n - 1$ verifications. However, we are using the efficient Rabin's signature scheme which requires only modulo multiplication for signature verification.

Merge Protocol

Our protocol requires only three constant rounds regardless of merging users m or groups k as compared to GDH, which scales linearly with the number of merging users m . For communication messages, our protocol, like STR and TGDH, is more efficient as it scales linearly with the number of merging groups k (which is usually very small) as compared to BD and GDH. Our protocol

only requires *two* users, U_1 and U_{n+1} , to perform five exponentiations each while all other users need not perform any exponentiations. The BD, however, requires every user to perform three exponentiations each. GDH is the least efficient as it is dependent on the group size and merging users. In terms of signature generation and verification, our protocol is the most efficient as *only* two users U_1 and U_{n+1} have to generate and verify *one* signature each while all other users need not generate or verify any signatures.

5.8 Discussion

In this section, we provide discussion and comparison of the existing GKA protocols and our proposed protocol when used in MANETs. Our proposed protocol incorporates autoconfiguration services with GKA while taking advantage of the *broadcast* nature of MANETs. As a comparison, each user U_i in the GDH protocol has to wait until it has received information from U_{i-1} before it can perform computations and transmit to U_{i+1} . Hence, communications in GDH are unicast and does not take full advantage of the broadcast capabilities in MANETs. If the autoconfiguration service was to be implemented in GDH, then all users have to wait for the controller U_n to compile and broadcast the IP address allocation table after $\mathcal{O}(n)$ communication rounds whereas in our proposed IKA protocol, each user can build up the IP address allocation table individually after two communication rounds. Furthermore, the controller U_n in GDH also has to perform the bulk of the $\mathcal{O}(n)$ computation in GKA. As most MANETs have no pre-existing infrastructures and all user nodes are usually low power devices; therefore there are no high power nodes to act as controller U_n in GDH. Besides, Kim et al [KPT04b] also highlighted that GDH does not efficiently handle network merge events. Hence, GDH might not be suitable for MANET application. Only in special MANET environments that have pre-existing infrastructure, where there are high power nodes, the GDH Leave protocol will be more efficient as the high power nodes can act as controller U_n .

Our proposed protocol uses a so called ring structure for GKA where users are arranged in a ring and the user after the last user U_n is again U_1 . The autoconfiguration service in our proposed protocol will generate an IP address allocation table/list, which can be used directly by higher layer MANET applications including routing protocols. The ring structure and IP address allocation table has to be stored after the initial key agreement for future dynamic events. It is rather

straightforward and efficient for users to combine both these two entities in our proposed protocol to lower the storage costs as compared to storing them separately. On the other hand, TGDH and STR require each user to store the entire tree structure, which is larger than the ring structure as it contains not only the users in the group, but also intermediate nodes that are parent nodes of two child nodes each. If TGDH or STR also uses autoconfiguration, it would be more complicated to combine the resulting IP address allocation table generated by the autoconfiguration service with the corresponding TGDH or STR tree structure, which contains the locations of the user nodes and intermediate nodes. One solution is to allow TGDH or STR that uses autoconfiguration services to generate an IP address allocation table that is of the same tree structure format and not in the usual list format; however this would mean that extra effort is required to convert these allocation tables back into the list format. Even if the MANET application (e.g. Tree-based Multicast protocol) uses a multicast tree, this tree is different from the key tree used by TGDH or STR as all nodes in the multicast tree denotes user nodes whereas in the TGDH or STR key tree, only leaf nodes are user nodes while intermediate nodes are virtual nodes that cannot contain user nodes. Therefore, the TGDH or STR key tree cannot be used as the multicast tree to save storage space.

The updating of the tree structure in TGDH and STR is also more complicated than our proposed protocol. For Join and Merge events, TGDH and STR may require some users, known as sponsor nodes, to create new nodes (leaves) in the tree and change its own position in the tree to accommodate the new user(s). Hence, all users in TGDH and STR will have to update not only the position of new user(s), but also the new position of affected users in the tree. Our proposed protocol only require the new user(s) to be appended to the bottom of the ring structure and the positions of current users are not affected. Although TGDH and STR requires fewer signature verifications than our proposed protocol in Leave events, however the tree structure in TGDH and STR changes as the leaving user(s) is/are deleted and some of the current users may have to take up new positions in the tree. Therefore, existing users have to re-structure their TGDH or STR tree by deleting leaving user(s) and also updating the positions of the existing affected users. Our proposed protocol however requires only the deletion of leaving users from the ring structure. It was mentioned in [SBL⁺04] that TGDH is suitable only for small groups. This is because the tree height h , which is directly proportional to the computational costs (Table 5.1), increases as group

size increases. Therefore, TGDH would be more efficient than our protocol only if the MANET group size is small enough (h is not too large) such that the overall costs of TGDH is lower.

STR was designed for wide-area networks (WANs) where the user nodes are workstations such as 850MHz Pentium III computers that do not have any power constraints and can perform large amount of cryptographic operations. STR is more efficient in WANs as it uses more cryptographic operations (computational costs) to reduce communications overhead, which is more expensive in WAN environment. However, user nodes in MANETs are usually battery operated low power devices such as PDAs with much slower processors, therefore both computational and communication costs are equally expensive for these nodes. As a result, it is not efficient to implement STR in MANETs since the use of more cryptographic operations would result in higher overall costs for user nodes. STR would be efficient only if it is implemented in special MANET environments where all user nodes are high power devices such as workstations that do not have any power constraints.

5.9 Summary

In this chapter, we first examined two methods to extend the BD GKA protocol to handle dynamic events such as user join and user leave. We showed that the Join method suggested by Steiner et al in [STW98] for the BD protocol does not provide backward secrecy, that is, new users can discover previous keys used. Next, we cryptanalyzed Dutta and Barua's provably secure Leave protocol [DB05] which is based on the BD protocol and show that a leaving user can still discover the new session key used in subsequent sessions if any two consecutive remaining users do not change their random secret elements (i.e. their proposed Leave Protocol does not provide forward secrecy). In the second part of this chapter, we proposed four protocols that combine secure IP address autoconfiguration, GKA and an efficient signature scheme to provide an efficient way of handling secure communications in MANET. The proposed protocols will change the group key and update the IP allocation table whenever dynamic events occur. This will ensure that subsequent communication sessions are protected from leaving users and previous communication sessions are protected from joining users. A complexity analysis also showed that despite having extra properties such as secure autoconfiguration, our proposed protocols are still comparatively efficient.

Chapter 6

Energy-Efficient ID-based Group Key Agreement Protocols for Wireless Networks

6.1 Introduction

In the previous chapter, we proposed authenticated dynamic GKA protocols to handle the dynamic group membership events such as user join, user leave and network merge which are often in wireless networks. The proposed GKA protocols in the previous chapter use certificate-based signature scheme to provide authentication. In order for users to verify signatures in a certificate-based signature scheme, the user needs to validate the signer's public key by verifying the digital certificate issued by Certification Authority (CA) before verifying the signature produced by the signer. This results in additional computational cost. ID-based signature schemes do not require the reception and verification of certificates as there are no requirements for public keys in ID-based schemes.

In this chapter, we present an authenticated Initial GKA (IKA) protocol for group formation and three authenticated dynamic protocols, namely Join, Leave and Merge protocol, to handle dynamic group membership events for symmetric wireless networks. The proposed protocols are based on the ID-based Guillou-Quisquater (GQ) signature scheme [GQ88] described in Section 2.3.4 and the BD protocol [BD95]. The GQ signature scheme is an efficient ID-based scheme that is not based on pairing, which has high computational complexity [STY04]. We compare our proposed protocols with other authenticated versions of the BD protocol and the ID-based Saeednia-Safavi-

Naini (SSN) GKA protocol [SSN98], which is also based on the BD protocol. Besides the normal complexity analysis, we also perform energy cost analysis based on the 133MHz “StrongARM” microprocessor and two communication transceivers commonly used in wireless networks to provide better understanding of the energy costs of user nodes when running GKA protocols in wireless networks. The complexity and energy analysis show that the energy consumed by nodes running the proposed IKA and dynamic protocols are lower in comparison.

6.2 Proposed ID-based Authenticated GKA Protocol

In this section, we first describe the proposed authenticated ID-based IKA protocol that is used during group formation to establish a common group key for all users in the group. The proposed IKA protocol uses batch verification based on the GQ ID-based signature scheme as described in Section 2.3.4 in Chapter 2. Next, we present three authenticated dynamic protocols, namely the Join, Merge and Leave protocol, that can be used to efficiently handle dynamic group membership events. Our proposed dynamic protocols use symmetric key cryptography, which studies such as [CKM00] and [HV02] have shown to have energy requirements of orders of magnitude lower than modular exponentiations.

6.2.1 Initial Group Key Agreement (IKA) Protocol

Let $\mathcal{G} = \{U_1, \dots, U_n\}$ be the initial group of n users where $U_i = ID_i$ for $1 \leq i \leq n$ refers to the 32-bit identity of user U_i . We choose 32-bits for the identity as 32-bits are sufficiently large enough to identify all users. We consider a so called ring structure among the users of $\mathcal{G}$ where the users' indices can be considered on the circulation of $\{1, \dots, n\}$. The proposed IKA protocol consist of two rounds during which each user U_i will broadcast their keying materials z_i and X_i in Round 1 and Round 2 respectively for group key computation. The IKA protocol is described as follows:

Setup : The Private Key Generator (PKG) selects two random 512 bits primes $\bar{p}$ and $\bar{q}$ and computes $\bar{N} = \bar{p}\bar{q}$ and $\Phi = (\bar{p} - 1)(\bar{q} - 1)$, where Φ is Euler's totient function. Next, The PKG selects an integer $e = 2^{16} + 1 = 65537$ and computes secret exponent d such that $ed = 1 \pmod{\Phi}$ for the GQ ID-based signature scheme. The PKG also selects two random large primes, q (160-bit) and p (1024-bit), such that q divides $p - 1$ for the GKA protocol. Next, an element $g \in Z_p^*$ of order q

is selected. Finally, the PKG selects a hash function $H : \{0, 1\}^* \rightarrow \{0, 1\}^\lambda$, where λ is an integer.

The parameters **params** are $(\bar{N}, e, p, q, g, H)$ and the master keys are $(\bar{p}, \bar{q}, d)$.

Extract : The PKG verifies $U_i = ID_i$ of user U_i and computes the secret key for U_i as $\mu_i = H(U_i)^d \bmod N$. The secret key μ_i is then sent securely to U_i .

Round 1 : Each user U_i for $1 \leq i \leq n$ first selects $r_i \in Z_q^*$ and $\tau_i \in Z_N^*$ and computes $z_i = g^{r_i} \bmod p$ and $t_i = \tau_i^e \bmod N$. U_i then broadcast $\{U_i || z_i || t_i\}$.

Round 2 : Each user U_i for $1 \leq i \leq n$ first computes :

$$X_i = \left(\frac{z_{i+1}}{z_{i-1}} \right)^{r_i} = g^{r_i r_{i+1} - r_{i-1} r_i} \bmod p \quad (6.1)$$

Next, U_i computes $\mathcal{Z} = \prod_{j=1}^n z_j \bmod p$, $\mathcal{T} = \prod_{j=1}^n t_j \bmod N$ and $\theta = H(\mathcal{T}, \mathcal{Z})$. U_i then computes $s_i = \tau_i \cdot \mu_i^\theta \bmod N$ and produces the signature $\sigma_i = (s_i, \theta)$. U_i stores $\mathcal{Z}$ and θ and broadcasts $\{U_i || X_i || s_i\}$. It is noted that U_1 is assumed to be a trusted controller such that U_1 will be the last user to broadcast its message after all the other users $U_j \neq U_1$ have broadcast their messages.

Authentication and Key Computation: Each U_i verifies the received signatures using the stored $\mathcal{Z}$ and θ to check the equation:

$$\theta = H \left(\left(\prod_{j=1}^n s_j \right)^e \cdot \left(\prod_{j=1}^n H(U_j) \right)^{-\theta}, \mathcal{Z} \right) \quad (6.2)$$

If equation (6.2) is correct, U_i will proceed to verify X_j for $1 \leq j \leq n$ broadcast in Round 2. However, if equation (6.2) is incorrect, then all members will retransmit again. From equation (6.1), we have $X_i = g^{r_i r_{i+1} - r_{i-1} r_i} \bmod p$ for $1 \leq i \leq n$, where $r_0 = r_n$ and $r_{n+1} = r_1$. Using Lemma 2 from Chapter 4, we can check whether X_j sent by U_j is genuine. If $\prod_{j=1}^n X_j \neq 1 \bmod p$, it means that at least one of the X_j is incorrect. Then, all members will retransmit again. If each U_i correctly verifies $\prod_{j=1}^n X_j = 1 \bmod p$, then U_i computes $g^{r_i r_{i+1}} = X_i \cdot g^{r_{i-1} r_i} \bmod p$ and recursively compute $g^{r_j r_{j+1}}$ using similar method as described in Lemma 3 case (i) and case (ii) for $1 \leq j \neq i \leq n$, where $r_{n+1} = r_1$. Hence, each U_i computes the group key K as follows:

$$K = \prod_{j=1}^n g^{r_j r_{j+1}} \bmod p = g^{r_1 r_2 + \dots + r_n r_1} \bmod p \quad (6.3)$$

where $r_0 = r_n$ and $r_{n+1} = r_1$.

6.2.2 Join Protocol

Let $\mathcal{G} = \{U_1, \dots, U_n\}$ be the current group and U_{n+1} be the new user joining the group. We divide $\mathcal{G}$ into two parts, $\{U_1, U_n\}$, which has the two users actively involved in the Join Protocol and $\{U_2, \dots, U_{n-1}\}$, which consists of the rest of the group. We consider a so called ring structure among the users of $\mathcal{G}$ with U_{n+1} joining $\mathcal{G}$ in between U_n and U_1 to form the new group $\mathcal{G}' = \{U_1, \dots, U_{n+1}\}$.

The Join Protocol consists of three rounds and is described as follows:

Round 1:

(i) The new node U_{n+1} first chooses random $r_{n+1} \in Z_q^*$ and computes $z_{n+1} = g^{r_{n+1}} \bmod p$. Next, U_{n+1} signs $U_{n+1}||z_{n+1}$ using the GQ ID-based signature scheme to obtain the signature σ_{n+1} and broadcasts $\{U_{n+1}||z_{n+1}||\sigma_{n+1}\}$.

(ii) U_n signs z_n using the GQ ID-based signature scheme and obtains signature σ'_n . Next, U_n sends $\{U_n||z_n||\sigma'_n\}$ to U_{n+1} .

Round 2: U_1 verifies the signature σ_{n+1} . If the signature is invalid, then U_1 rejects U_{n+1} . Otherwise, U_1 chooses new random $r'_1 \in Z_q^*$ and computes K^* as follows:

$$\begin{aligned} K^* &= K \cdot (z_2 \cdot z_n)^{-r_1} \cdot (z_2 \cdot z_{n+1})^{r'_1} \bmod p \\ &= g^{r'_1 r_2 + r_2 r_3 + \dots + r_{n-1} r_n + r_{n+1} r'_1} \bmod p. \end{aligned} \quad (6.4)$$

U_1 then encrypts $K^*||U_1$ using the previously established group key K and a symmetric key encryption SKE to obtain $c_1 = \mathcal{E}_K(K^*||U_1)$ and broadcasts $\{U_1||c_1\}$ to existing group $\mathcal{G}$.

Round 3 : U_n verifies signature σ_{n+1} . If the signature is invalid, then U_n rejects U_{n+1} . Otherwise, U_n computes DH key $K_{U_n U_{n+1}} = z_{n+1}^{r_n} = g^{r_n r_{n+1}} \bmod p$, which it shares with U_{n+1} . Next, U_n encrypts $K_{U_n U_{n+1}}||U_n$ using previously established group key K to obtain $c_n = \mathcal{E}_K(K_{U_n U_{n+1}}||U_n)$. U_n then decrypts c_1 and obtains K^* and identity U_1 . If the identity U_1 is not decrypted correctly, then U_n rejects c_1 . Otherwise, U_n encrypts $K^*||U_n$ using $K_{U_n U_{n+1}}$ to obtain $c'_n = \mathcal{E}_{K_{U_n U_{n+1}}}(K^*||U_n)$. Finally U_n broadcasts $\{U_n||c_n||c'_n\}$.

Key Computation :

- (i) U_{n+1} first verifies signature σ'_n . If the signature is invalid, then U_{n+1} rejects U_n . Otherwise, U_{n+1} computes $K_{U_n U_{n+1}} = z_n^{r_{n+1}} = g^{r_n r_{n+1}} \bmod p$ and decrypts c'_n to obtain K^* and the identity U_n . If the identity U_n is not decrypted correctly, then U_{n+1} rejects c'_n . Otherwise, U_{n+1} accepts K^* .
- (ii) Users U_i for $2 \leq i \leq n-1$ decrypts c_1 using the previously established group key K and obtains K^* and the identity U_1 . If the identity U_1 is not decrypted correctly, then U_i rejects c_1 . Otherwise, U_i accepts K^* . Next, users U_i for $1 \leq i \leq n-1$ decrypts c_n using the previously established group key K and obtains $K_{U_n U_{n+1}}$ and the identity U_n . If the identity U_n is not decrypted correctly, then U_i rejects c_n . Otherwise, U_i accepts $K_{U_n U_{n+1}}$.
- (iii) Finally, all users U_i for $1 \leq i \leq n$ including U_{n+1} compute the new group key K' as follows:

$$\begin{aligned} K' &= K^* \cdot K_{U_n U_{n+1}} \bmod p \\ &= g^{r'_1 r_2 + r_2 r_3 + \dots + r_n r_{n+1} + r_{n+1} r'_1} \bmod p \end{aligned} \quad (6.5)$$

6.2.3 Merge Protocol

A merge occurs when two or more subgroups are combined into a single group. Let $\mathcal{G}_A = \{U_1, U_2, \dots, U_n\}$ denote all users in Group $\mathcal{G}_A$ with previously established group key K_A and $\mathcal{G}_B = \{U_{n+1}, U_{n+2}, \dots, U_{n+m}\}$ denote all users in Group $\mathcal{G}_B$ with previously established group key K_B . We divide $\mathcal{G}_A$ into $\{U_1, U_n\}$ and $\{U_2, \dots, U_{n-1}\}$ and $\mathcal{G}_B$ into $\{U_{n+1}, U_{n+m}\}$ and $\{U_{n+2}, \dots, U_{n+m-1}\}$. We consider a so called ring structure for group $\mathcal{G}_A$, group $\mathcal{G}_B$ and the merged group $\mathcal{G}' = \mathcal{G}_A \cup \mathcal{G}_B = \{U_1, \dots, U_{n+m}\}$. The Merge Protocol consists of three rounds and is described as follows:

Round 1 :

- (i) U_1 first selects new random $r'_1 \in Z_q^*$ and computes $z'_1 = g^{r'_1} \bmod p$. Next, U_1 signs $U_1 || z'_1 || z_n$ using GQ ID-based signature scheme to produce signature σ'_1 and broadcasts $\{U_1 || z'_1 || z_n || \sigma'_1\}$.
- (ii) U_{n+1} selects new random $r'_{n+1} \in Z_q^*$, computes $z'_{n+1} = g^{r'_{n+1}} \bmod p$ and signs $U_{n+1} || z'_{n+1} || z_{n+m}$ using GQ ID-based signature scheme to obtain signature σ'_{n+1} and broadcasts $\{U_{n+1} || z'_{n+1} || z_{n+m} || \sigma'_{n+1}\}$.

Round 2 :

- (i) U_1 verifies the signature σ'_{n+1} . If the signature is invalid, then U_1 rejects U_{n+1} . Otherwise, U_1 computes $K_{U_1 U_{n+1}} = (z'_{n+1})^{r'_1} = g^{r'_1 r'_{n+1}} \bmod p$, the DH key shared with U_{n+1} . U_1 then computes

K_A^* as follows:

$$\begin{aligned} K_A^* &= K_A \cdot (z_2 \cdot z_n)^{-r_1} \cdot (z_2 \cdot z_{n+m})^{r'_1} \\ &= g^{r'_1 r_2 + r_2 r_3 + \dots + r_{n-1} r_n + r_{n+m} r'_1} \mod p \end{aligned} \quad (6.6)$$

Next, U_1 encrypts $K_A^* || U_1$ using group $\mathcal{G}_A$'s previously established key K_A and $K_{U_1 U_{n+1}}$ to obtain $c_1 = \mathcal{E}_{K_A}(K_A^* || U_1)$ and $c'_1 = \mathcal{E}_{K_{U_1 U_{n+1}}}(K_A^* || U_1)$ respectively. Lastly, U_1 broadcasts $\{U_1 || c_1 || c'_1\}$.

(ii) U_{n+1} first verifies the signature σ'_1 . If the signature is invalid, then U_{n+1} rejects U_1 . Otherwise, then U_{n+1} computes $K_{U_1 U_{n+1}} = (z'_1)^{r'_{n+1}} = g^{r'_1 r'_{n+1}} \mod p$, the DH key shared with U_1 . U_{n+1} then computes K_B^* as follows:

$$\begin{aligned} K_B^* &= K_B \cdot (z_n \cdot z_{n+2})^{r'_{n+1}} \cdot (z_{n+2} \cdot z_{n+m})^{-r_{n+1}} \\ &= g^{r_n r'_{n+1} + r'_{n+1} r_{n+2} + r_{n+2} r_{n+3} + \dots + r_{n+m-1} r_{n+m}} \mod p \end{aligned} \quad (6.7)$$

Next, U_{n+1} encrypts $K_B^* || U_{n+1}$ using group $\mathcal{G}_B$'s previously established key K_B and $K_{U_1 U_{n+1}}$ to obtain $c_{n+1} = \mathcal{E}_{K_B}(K_B^* || U_{n+1})$ and $c'_{n+1} = \mathcal{E}_{K_{U_1 U_{n+1}}}(K_B^* || U_{n+1})$ respectively. Finally, U_{n+1} broadcasts $\{U_{n+1} || c_{n+1} || c'_{n+1}\}$.

Round 3 :

(i) U_1 decrypts c'_{n+1} to obtain K_B^* and the identity U_{n+1} . If the identity U_{n+1} is not decrypted correctly, then U_1 rejects c'_{n+1} . Otherwise, U_1 accepts K_B^* . U_1 then encrypts $K_B^* || U_1$ using group $\mathcal{G}_A$'s previously established group key K_A to obtain $c''_1 = \mathcal{E}_{K_A}(K_B^* || U_1)$. Finally, U_1 broadcasts $\{U_1 || c''_1\}$ to group $\mathcal{G}_A$.

(ii) U_{n+1} decrypts c'_1 to obtain K_A^* and the identity U_1 . If the identity U_1 is not decrypted correctly, then U_{n+1} rejects c'_1 . Otherwise, U_{n+1} accepts K_A^* . U_{n+1} then encrypts $K_A^* || U_{n+1}$ using group $\mathcal{G}_B$'s previously established group key K_B to obtain $c''_{n+1} = \mathcal{E}_{K_B}(K_A^* || U_{n+1})$. Finally, U_{n+1} broadcasts $\{U_{n+1} || c''_{n+1}\}$ to group $\mathcal{G}_B$.

Key Computation :

(i) Group $\mathcal{G}_A$ users U_i for $2 \leq i \leq n$ first decrypts c_1 using previously established group key K_A and obtains K_A^* and the identity U_1 . If the identity U_1 is not decrypted correctly, then U_i rejects c_1 . Otherwise, U_i accepts K_A^* . Next, U_i decrypts c''_1 using previously established group key K_A and

obtains K_B^* and the identity U_1 . If the identity U_1 is not decrypted correctly, then U_i rejects c_1'' . Otherwise, U_i accepts K_B^* .

(ii) Group $\mathcal{G}_B$ users U_j for $n+2 \leq j \leq n+m$ first decrypts c_{n+1} using previously established group key K_B and obtains K_B^* and the identity U_{n+1} . If the identity U_{n+1} is not decrypted correctly, then U_j rejects c_{n+1} . Otherwise, U_j accepts K_B^* . Next, U_j decrypts c_{n+1}'' using previously established group key K_B and obtains K_A^* and the identity U_{n+1} . If the identity U_{n+1} is not decrypted correctly, then U_j rejects c_{n+1}'' . Otherwise, U_j accepts K_A^* .

(iii) Finally, all users in the merged group $\mathcal{G}'$ compute the new group key K' as follows:

$$\begin{aligned} K' &= K_A^* \cdot K_B^* \bmod p \\ &= g^{r_1' r_2 + r_2 r_3 + \dots + r_n r_{n+1}' + r_{n+1}' r_{n+2} + r_{n+2} r_{n+3} + \dots + r_{n+m} r_1'} \bmod p. \end{aligned} \quad (6.8)$$

6.2.4 Leave Protocol

Let $\mathcal{G} = \{U_1, \dots, U_n\}$ denote the previously established group, U_l denote the leaving user and $\mathcal{G}' = \mathcal{G} \setminus \{U_l\}$ denotes the new group. We consider a so called ring structure for $\mathcal{G}$ and $\mathcal{G}'$. The Leave Protocol comprises of two rounds and is described as follows:

Round 1 :

(i) All remaining *odd-indexed* users $U_j \neq U_l$ for j is odd select new randoms $r_j' \in Z_q^*$ and $\tau_j' \in Z_N^*$ and computes $z_j' = g^{r_j'} \bmod p$ and $t_j' = (\tau_j')^e \bmod N$. U_j then broadcasts $\{U_j || z_j' || t_j'\}$.

(ii) All remaining *even-indexed* users $U_j \neq U_l$ for j is even select new random $\tau_j' \in Z_N^*$ and computes $t_j' = (\tau_j')^e \bmod N$. U_j then broadcasts $\{U_j || t_j'\}$.

Round 2 : All remaining users $U_i \in \mathcal{G}'$ for $1 \leq i \neq l \leq n$ compute X_i' as follows:

$$X_i' = \begin{cases} \left(\frac{z_{i+1}'}{z_{i-1}'} \right)^{r_i'} & \text{if } i \neq l, l-1, l+1, \\ \left(\frac{z_{l+1}'}{z_{l-2}'} \right)^{r_{l-1}'} & \text{if } i = l-1, \\ \left(\frac{z_{l+2}'}{z_{l-1}'} \right)^{r_{l+1}'} & \text{if } i = l+1. \end{cases} \quad (6.9)$$

where $z_i' = z_i$ and $r_i' = r_i$ for i is even. Next, each $U_i \in \mathcal{G}'$ for $1 \leq i \neq l \leq n$ computes $\mathcal{Z}' = \prod_{j=1, j \neq l}^n z_j' \bmod p$, $\mathcal{T}' = \prod_{j=1, j \neq l}^n t_j' \bmod N$ and $\theta' = H(\mathcal{T}', \mathcal{Z}')$. $U_i \in \mathcal{G}'$ for $1 \leq i \neq l \leq n$ then computes $s_i' = \tau_i' \cdot \mu_i^{\theta'}$ and produces the signature $\sigma_i' = (s_i', \theta')$. $U_i \in \mathcal{G}'$ for $1 \leq i \neq l \leq n$

stores $\mathcal{Z}'$ and θ' and broadcasts $\{\mathbf{U}_i || X'_i || s'_i\}$. $\mathbf{U}_1$ is assumed to be a trusted controller such that $\mathbf{U}_1$ will be the last user to broadcast its message after all the other users have broadcast their messages. It is assumed that $\mathbf{U}_1$ never leaves the group.

Authentication and Key Computation: Each $\mathbf{U}_i \in \mathcal{G}'$ for $1 \leq i \neq l \leq n$ verifies the received signatures using the stored $\mathcal{Z}'$ and θ' to check the equation:

$$\theta' = \text{H} \left(\left(\prod_{j=1, j \neq l}^n s'_j \right)^e \cdot \left(\prod_{j=1, j \neq l}^n \text{H}(\mathbf{U}_j) \right)^{-\theta'} \right), \mathcal{Z}' \quad (6.10)$$

If equation (6.10) is correct, each $\mathbf{U}_i \in \mathcal{G}'$ for $1 \leq i \neq l \leq n$ proceeds to verify $\prod_{j=1, j \neq l}^n X'_j = 1 \bmod p$. If $\prod_{j=1, j \neq l}^n X'_j \neq 1 \bmod p$, then at least one of the X'_j is incorrect and all members will retransmit again. If each $\mathbf{U}_i \in \mathcal{G}'$ for $1 \leq i \neq l \leq n$ correctly verifies that $\prod_{j=1, j \neq l}^n X'_j = 1 \bmod p$, then from equation (6.9) and using similar calculations as mentioned in the Authentication and Key Computation Round of the proposed IKA protocol in Section 6.2.1, each $\mathbf{U}_i \in \mathcal{G}'$ for $1 \leq i \neq l \leq n$ obtains $g^{r'_j r'_{j+1}} \bmod p$ for $1 \leq j \neq l \leq n$, where $r'_j = r_j$ for j is even. Then each $\mathbf{U}_i \in \mathcal{G}'$ for $1 \leq i \neq l \leq n$ computes the new group key K' as follows:

$$K' = \prod_{j=1, j \neq l}^n g^{r'_j r'_{j+1}} = g^{r'_1 r'_2 + r'_2 r'_3 + \dots + r'_{l-2} r'_{l-1} + r'_{l-1} r'_{l+1} + \dots + r'_n r'_1} \bmod p \quad (6.11)$$

where $r'_j = r_j$ for j is even.

6.3 Security Analysis

In this section, we provide security analysis for our proposed protocols, namely the IKA, Join, Merge and Leave protocols. We show that our protocols are secure as follows:

6.3.1 Security Analysis of IKA Protocol

In this section, we prove that an adversary $\mathcal{A}$ is not able to obtain the group key K from the proposed IKA protocol and the IKA is secure against insider impersonation.

Theorem 16 *An adversary $\mathcal{A}$ is unable to obtain the group key K in the proposed IKA protocol provided the GQ ID-based signature scheme and the BD protocol are secure.*

Proof: Suppose an adversary $\mathcal{A}$ tries to impersonate a group member U_i (for some i) to obtain a group key, $\mathcal{A}$ first generates randoms $z_i = g^{r_i} \bmod p$ and $t_i = \tau_i^e \bmod N$ in Round 1, then in Round 2, $\mathcal{A}$ has to compute the signature $\sigma_i = (s_i, \theta)$ where $s_i = \tau_i \cdot \mu_i^\theta$, μ_i is the user U_i private key and $\theta = H(\prod_{j=1}^n t_j, \prod_{j=1}^n z_j)$. The validity of σ_i can be checked through the equation (6.2) by the other group members when they received all signatures. If the equation (6.2) is incorrect, then the group member will know that some of the user is not a valid user. As $\mathcal{A}$ does not know the secret key μ_i of U_i and the GQ ID-based signature scheme is assumed to be secure, therefore, $\mathcal{A}$ is not able to produce a valid signature σ_i . Hence, the adversary $\mathcal{A}$ is not able to impersonate the user U_i to establish a group key. Now, consider that an adversary $\mathcal{A}$ can only obtain the public information from the group members sent. Then, this is just a BD protocol by ignoring the signature (σ_j for $1 \leq j \leq n$) in round 2 and t_j (for $1 \leq j \leq n$) in round 1. As BD protocol is assumed to be secure, hence, the adversary $\mathcal{A}$ is not able to obtain the group key established by a group. ■

Theorem 17 *The proposed IKA protocol is secure against insider impersonation if the GQ ID-based signature scheme is secure.*

Proof: Assume that there are k ($< n - 1$) group members in a group of n members impersonating a user U_i (for some i) to establish a new group key, then there is at least one group member, say U_j (who is not one of the k impersonators) sending a new z'_j in round 1 which is different from that of the previously sent. Therefore, $\mathcal{Z}' = \prod_{k=1}^n z'_k$ will be different from the previous $\mathcal{Z}$ sent. In this case, the k number of impersonators need to produce a valid signature σ'_i (which makes a commitment of t_i in round 1) of the message $\mathcal{Z}'$ for the user U_i in round 2. But, this is impossible for the k number of impersonators to produce such valid signature as they do not know the private key of U_i and the GQ ID-based signature scheme is assumed to be secure. Therefore, it is not possible for any k ($< n - 1$) insiders to impersonate any user in the group to establish a new group key. ■

6.3.2 Security Analysis of Join Protocol

In this section, we first show that the Join Protocol achieves backward secrecy, that is, the new user U_{n+1} cannot obtain the previous group key K from new group key K' with the information known by the new user U_{n+1} . We then show that an adversary $\mathcal{A}$ cannot obtain the new group key K' from the proposed Join protocol.

Theorem 18 *The proposed Join protocol achieves backward secrecy in the information-theoretic sense.*

Proof: Suppose new user U_{n+1} tries to obtain the previous group key $K = g^{r_1 r_2 + r_2 r_3 + \dots + r_{n-1} r_n + r_n r_1}$ from the Join Protocol, then U_{n+1} first obtained $K^* = g^{r'_1 r_2 + r_2 r_3 + \dots + r_{n-1} r_n + r_n r_{n+1} r'_1}$ and a new group key $K' = g^{r'_1 r_2 + r_2 r_3 + \dots + r_{n-1} r_n + r_n r_{n+1} + r_{n+1} r'_1}$ from the Join protocol. The difference between K and K^* are the terms $g^{r_1 r_2}$ and $g^{r'_1 r_2}$; and $g^{r_n r_1}$ and $g^{r_{n+1} r'_1}$ respectively. As r'_1 and r_1 are random and independent to each other, therefore, $g^{r'_1 r_2}$ and $g^{r_1 r_2}$ are independent to each other. Similarly, $g^{r_{n+1} r'_1}$ and $g^{r_n r_1}$ are independent to each other. Therefore, K and K^* will look random and independent to each other. Furthermore, there is no information about r'_1 being send out except in K^* and K' is obtained by multiplying K^* with $g^{r_n r_{n+1}}$. Hence, the new user will not be able to derive K from K^* and K' . Therefore, we conclude that the Join protocol achieves backward secrecy in the information-theoretic sense. $\blacksquare$

Next, we show that the outsider malicious adversary $\mathcal{A}$ is unable to obtain the new group key K' from the Join Protocol.

Theorem 19 *An adversary $\mathcal{A}$ is not able to obtain the new group key K' from the proposed Join Protocol provided the GQ ID-based signature scheme and the symmetric key encryption scheme SKE are secure and the CDH problem is difficult.*

Proof: In this proof, we consider two cases, that is, (a) an adversary $\mathcal{A}$ tries to impersonate a user U_i (for some $i = n$ or $i = n + 1$) to establish a new group key, (b) an adversary $\mathcal{A}$ tries to obtain a new group key from the public exchange information. For case (a), $\mathcal{A}$ has to compute the signature σ'_n (σ_{n+1}) in Round 1 if $\mathcal{A}$ wants to impersonate U_n (U_{n+1}). The validity of σ_{n+1} is checked by U_1 and U_n in Round 2 and 3 respectively (the validity of σ'_n is checked by U_{n+1} in

the Key Computation Round). As $\mathcal{A}$ does not know the secret key μ_i of $\mathbf{U}_i$ and the GQ ID-based signature scheme is assumed to be secure, hence $\mathcal{A}$ is not able to produce the signature σ'_n (σ_{n+1}). Therefore, it is not possible for an adversary $\mathcal{A}$ to obtain the new group key in this case. Now, we consider the case (b) as follows: As an adversary $\mathcal{A}$ is not a joining new user $\mathbf{U}_{n+1}$, $\mathcal{A}$ can only obtain the public exchange information, that is, z_n , z_{n+1} , $\mathbf{c}_1$, $\mathbf{c}_n$ and $\mathbf{c}'_n$. As the new group key $K' = K^* \cdot g^{r_n r_{n+1}}$ where K^* and $g^{r_n r_{n+1}}$ are the plaintext of $\mathbf{c}_1$ and $\mathbf{c}_n$ respectively, therefore, it is not possible for $\mathcal{A}$ to recover K^* and $g^{r_n r_{n+1}}$ from $\mathbf{c}_1$ and $\mathbf{c}_n$ respectively as it is assumed that the symmetric key encryption scheme SKE is secure. On the other hand, if the adversary $\mathcal{A}$ can obtain $g^{r_n r_{n+1}}$ from z_n and z_{n+1} , then the adversary $\mathcal{A}$ can obtain K^* from $\mathbf{c}'_n$ and $\mathcal{A}$ obtains the new group key $K' = K^* \cdot g^{r_n r_{n+1}}$. But this is not possible for the adversary $\mathcal{A}$ to obtain $g^{r_n r_{n+1}}$ as this is the CDH problem and is assumed to be difficult. Therefore, the adversary $\mathcal{A}$ is not possible to obtain the new group key K' . ■

6.3.3 Security Analysis of Merge Protocol

In this section, we first show that the Merge Protocol provides backward secrecy, that is, users in group $\mathcal{G}_B$ cannot obtain group $\mathcal{G}_A$'s previous group key K_A and users in group $\mathcal{G}_A$ cannot obtain group $\mathcal{G}_B$'s previous key K_B .

Theorem 20 *The proposed Merge protocol achieves backward secrecy provided CDH problem is difficult.*

Proof: In this theorem, we need to show that any user in group $\mathcal{G}_B$ ($\mathcal{G}_A$) is not able to obtain the previous group key K_A (K_B) of group $\mathcal{G}_A$ ($\mathcal{G}_B$). Suppose a user $\mathbf{U}_k$ ($n+1 \leq k \leq n+m$) in group $\mathcal{G}_B$ tries to obtain the previous group key $K_A = g^{r_1 r_2 + r_2 r_3 + \dots + r_{n-1} r_n + r_n r_1}$ of group $\mathcal{G}_A$ from the Merge protocol, then $\mathbf{U}_{n+1}$ first obtained z'_1 and $K_A^* = g^{r'_1 r_2 + r_2 r_3 + \dots + r_{n-1} r_n + r_n r'_1}$ from the merge protocol. The difference between K_A and K_A^* are the terms $g^{r_1 r_2}$ and $g^{r'_1 r_2}$; and $g^{r_n r_1}$ and $g^{r_{n+m} r'_1}$ respectively. As r'_1 and r_1 are random and independent to each other, therefore, $g^{r'_1 r_2}$ and $g^{r_1 r_2}$ are independent to each other. Similarly, $g^{r_{n+m} r'_1}$ and $g^{r_n r_1}$ are independent to each other. Therefore, K_A and K_A^* will look random and independent to each other. Furthermore, if $\mathbf{U}_k$ can obtain $g^{r_n r_1}$, $g^{r_1 r_2}$, $g^{r_{n+m} r'_1}$ (except user $\mathbf{U}_{n+m}$ which can compute this as he knows r_{n+m}) and $g^{r'_1 r_2}$ from z_n , z_1 , z'_1 , z_{n+m} and z_2 . Then, $\mathbf{U}_k$ can solve the CDH problem, but this contradicts to the

assumption that the CDH problem is difficult. Hence, the user U_k in group $\mathcal{G}_B$ will not be able to derive K_A from K_A^* . The similar proof can be shown that any user in group $\mathcal{G}_A$ is not able to obtain the previous group key K_B of group $\mathcal{G}_B$. Therefore, we conclude that the Merge protocol achieves backward secrecy if the CDH problem is difficult. ■

Next, we show that an adversary $\mathcal{A}$ is not able to obtain the new group key K' from the Merge Protocol.

Theorem 21 *An adversary $\mathcal{A}$ is not able to obtain the new group key K' from the proposed Merge Protocol provided the GQ ID-based signature scheme and the symmetric key encryption scheme SKE are secure and the CDH problem is difficult.*

Proof: In this proof, we consider two cases, that is, (a) an adversary $\mathcal{A}$ tries to impersonate a user U_i (for some $i = 1$ or $i = n + 1$) to establish a new group key, (b) an adversary $\mathcal{A}$ tries to obtain a new group key from the public exchange information. For case (a), $\mathcal{A}$ has to compute the signature σ'_1 (σ'_{n+1}) in Round 1 if $\mathcal{A}$ wants to impersonate U_1 (U_{n+1}). The validity of σ'_1 (σ'_{n+1}) is checked by U_{n+1} (U_1) respectively in Round 2. As $\mathcal{A}$ does not know the secret key μ_i of U_i and the GQ ID-based signature scheme is assumed to be secure, hence $\mathcal{A}$ is not able to produce the signature σ'_1 (σ'_{n+1}). Therefore, it is not possible for an adversary to obtain the new group key in this case. Now, we consider the case (b) as follows: As an adversary $\mathcal{A}$ is neither a legitimate user in group $\mathcal{G}_A$ nor group $\mathcal{G}_B$, $\mathcal{A}$ can only obtain the public exchange information, that is, z'_1 , z_n , z'_{n+1} , z_{n+m} , c_1 , c'_1 , c''_1 , c_{n+1} , c'_{n+1} and c''_{n+1} . As the new group key $K' = K_A^* \cdot K_B^*$ where K_A^* is the plaintext of c_1 and c''_{n+1} and K_B^* is the plaintext of c_{n+1} and c'_1 respectively, therefore, it is not possible for $\mathcal{A}$ to recover K_A^* from c_1 and c''_{n+1} and K_B^* from c_{n+1} and c'_1 respectively as it is assumed that the symmetric key encryption scheme SKE is secure. On the other hand, if the adversary $\mathcal{A}$ can obtain $g^{r'_1 r'_{n+1}}$ from z'_1 and z'_{n+1} , then the adversary $\mathcal{A}$ can obtain K_A^* and K_B^* from c'_1 and c'_{n+1} respectively and $\mathcal{A}$ obtains the new group key $K' = K_A^* \cdot K_B^*$. But this is not possible for the adversary $\mathcal{A}$ to obtain $g^{r'_1 r'_{n+1}}$ as this is the CDH problem and is assumed to be difficult. Therefore, the adversary $\mathcal{A}$ is not possible to obtain the new group key K' . ■

6.3.4 Security Analysis of Leave Protocol

In this section, we first show that the proposed Leave protocol achieves forward secrecy, that is, the leaving user U_l cannot obtain the previous group key K from new group key K' with the information known by the leaving user U_l . We then show that an adversary $\mathcal{A}$ cannot obtain the new group key K' from the proposed Leave protocol.

Theorem 22 *The proposed Leave protocol achieves forward secrecy provided the CDH problem is difficult and the BD protocol is secure.*

Proof: As the leaving user U_l knows the previous group key $K = g^{r_n r_1} \cdot \prod_{i=1}^{n-1} g^{r_i r_{i+1}}$ (which is of n terms of the form g^{rs}) and all $g^{r_i r_{i+1}}$ for $1 \leq i \leq n-1$ and $g^{r_n r_1}$, but the new group key is of the form $K' = g^{r'_1 r'_2 + r'_2 r'_3 + \dots + r'_{l-2} r'_{l-1} + r'_{l-1} r'_{l+1} + \dots + r'_n r'_1}$ (which is of $n-1$ terms of the form g^{rs}) where $r'_j = r_j$ for $1 \leq j \neq l \leq n$ and j is even. Furthermore, each r'_j is independent of r_j for $1 \leq j \neq l \leq n$ and j is odd, therefore, $g^{r'_j r_{j+1}}$ is independent of $g^{r_j r_{j+1}}$. On the other hand, if the leaving user U_l can compute $g^{r'_j r_{j+1}}$ from $g^{r'_j}$ and $g^{r_{j+1}}$ which is the CDH problem, then this contradicts to the fact that the CDH problem is difficult. Furthermore, the two terms $g^{r'_{l-2} r'_{l-1}}$ and $g^{r'_{l-1} r'_{l+1}}$ which are only appeared in K' , but not in K . Therefore, K' is independent of K . Hence, even the leaving user U_l knows all $g^{r_i r_{i+1}}$ for $1 \leq i \leq n-1$ and $g^{r_n r_1}$, it is not possible for U_l to obtain a new group key K' from K . This leads to the case that the leaving user U_l can only derive the new group key K' through the public exchange information, that is, round 1 and round 2 information sent by the remaining group members. But the Leave protocol (without the signature information sent) is just the BD protocol which is assumed to be secure, hence, the leaving user U_l is not able to obtain the new group key K' . Therefore, the Leave protocol achieves forward secrecy. ■

Theorem 23 *An adversary $\mathcal{A}$ (not a leaving user U_l) cannot obtain the new group key K' in the proposed Leave protocol provided that the BD protocol and the GQ ID-based signature scheme are secure.*

Proof: In this proof, we consider two cases, that is, (a) an adversary $\mathcal{A}$ tries to impersonate a user U_i (for some i is odd such that $i \neq l$) to establish a new group key, (b) an adversary $\mathcal{A}$ tries to obtain a new group key from the public exchange information. For case (a), it is already shown

in Theorem 16 that no adversary can impersonate any user in the group due to the secure GQ ID-based signature scheme. Therefore, it is not possible for an adversary to obtain the new group key in this case. Now, we consider the case (b) as follows: As an adversary $\mathcal{A}$ is not a leaving user U_l , $\mathcal{A}$ can only obtain the public exchange information, that is, z_i, X_i for $1 \leq i \leq n$ and z'_j, X'_j for $1 \leq j \neq l \leq n$ from IKA protocol and the Leave protocol respectively. It is noted that z'_i and z_i are independent to each other for i is odd and $i \neq l$ and X'_i and X_i are independent to each other for $1 \leq i \leq n$ such that $i \neq l, l-1, l+1$. Furthermore, the number of terms of the form g^{rs} in K and K' are different by 1. Therefore, the key K and K' is independent to each other. Hence, the adversary $\mathcal{A}$ can only derive the key K' from the public exchange information in the Leave protocol (without the signature information sent) which is just the BD protocol. But, the BD protocol is assumed to be secure, therefore, the adversary $\mathcal{A}$ is not able to obtain the new group key K' from the public exchange information. ■

6.4 Complexity Analysis

In this section, we first analyze and compare the computational (number of modular exponentiations, number of certificate verifications, number of signature generation and verifications and number of Map-To-Point operations) and communication (number of messages and certificate sent and received) costs of our proposed IKA protocol with three variants of the authenticated BD protocol and the Saeednia-Safavi-Naini (SSN) scheme. Next, we compare the computational and communication costs of our proposed dynamic protocols, namely Join, Merge and Leave protocols, with the authenticated BD protocol using the efficient certificate-based ECDSA signature scheme. As mentioned earlier, the BD protocol is re-executed whenever dynamic group membership events occur as no dynamic protocols have been specified by Burmester and Desmedt in [BD95]. The complexity of the BD protocol is also presented in [AKNR⁺04] and [KPT04b]. The current group size, merging users, merging groups and remaining *odd-indexed* users are denoted as n, m, k and v respectively.

IKA Protocol Comparison

Table 6.1 presents the complexity analysis of different protocols to achieve authenticated BD and the Saeednia-Safavi-Naini (SSN) scheme [SSN98]. The protocol in the first column of Table

6.1 refers to our proposed GKA protocol in Section 6.2.1. Protocol 2 uses 192-bit¹ ID-based Sakai et al's SOK signature scheme [SOK00] to provide authentication for BD. Protocols 3 and 4 are the BD with 163-bit ECDSA and BD with 1024-bit DSA signature scheme respectively. The last protocol, 1024-bit SSN scheme, uses ID-based cryptography. Although the last protocol does not require any signature generation and verifications, the number of modular exponentiations required is $2n + 4$.

Table 6.1: Complexity Analysis for Authenticated BD Protocol

	Proposed IKA Protocol	Protocol 2 (BD w/ SOK)	Protocol 3 (BD w/ ECDSA)	Protocol 4 (BD w/ DSA)	SSN scheme
Exponentiation	3	3	3	3	$2n + 4$
Message Transmission	2	2	2	2	2
Message Reception	$2(n - 1)$	$2(n - 1)$	$2(n - 1)$	$2(n - 1)$	$2(n - 1)$
Certificate Transmission	-	-	1	1	-
Certificate Reception	-	-	$n - 1$	$n - 1$	-
Certificate Verification	-	-	$n - 1$	$n - 1$	-
Map-To-Point	-	$n - 1$	-	-	-
Signature Generation	1	1	1	1	-
Signature Verification	1	$n - 1$	$n - 1$	$n - 1$	-

All other protocols require only three modular exponentiations for the BD protocol. All protocols, including SSN scheme, require two message transmission and $2(n - 1)$ messages to be received by each user. Protocols 3 and 4 require each user to transmit their certificate in the first message as well as receive and verify $n - 1$ certificates from other users. Protocol 2 is based on pairing and requires $n - 1$ MapToPoint operations.

All protocols except the SSN scheme requires one signature generation. This signature generation is done in Round 2 of the GKA for Protocol 2, 3 and 4 where each user signs the message $\{\mathbf{u}_i || z_i || X_i || \prod_{i=1}^n z_i\}$ to provide authentication for both keying materials z_i and X_i broadcast in Rounds 1 and 2 of the BD respectively. In terms of signature verifications, our proposed protocol is the most efficient.

Dynamic Protocols Comparison

Table 6.2 shows the complexity analysis of the proposed dynamic protocols, namely Join,

¹We assume that the NIST P-192 recommended elliptic curve over F_p is used in the SOK signature scheme.

Merge and Leave protocols, with the authenticated BD protocol using the efficient certificate-based ECDSA signature scheme. For communication costs, we focus on the number of rounds and the total number of messages. For computational costs, we focus on the number of exponentiations, signature generations, and signature verifications to be performed by each user.

Table 6.2: Complexity Analysis of Dynamic Protocols

Protocol		Communication Costs		Computational Costs		
		Rounds	Messages	Mod. Exp.	Signature Generation	Signature Verification
BD [BD95]	Join	2	$2n + 2$	3^a	2	$n + 2$
	Leave	2	$2n - 2$	3^a	2	n
	Merge	2	$2n + 2m$	3^a	2	$n + m + 1$
Proposed Protocols	Join	3	5	2^b	1	1
	Leave	2	$v + n - 2$	3^c	1	1
	Merge	3	$6(k - 1)$	4^d	1	1

a : All users in BD perform 3 exponentiations.

b : *Only* U_1 and U_{n+1} perform 2 exponentiations each.

c : *Only* U_j for j is odd perform 3 exponentiations and the rest performs 2 exponentiations each.

d : *Only* U_1 and U_{n+1} perform 4 exponentiations each.

Although our Join and Merge Protocols require 1 extra round than the BD Join and Merge Protocol, however both our Join and Merge Protocols require fewer communication messages than the BD Join and Merge Protocols. For exponentiation costs, all users in the BD perform 3 exponentiations each for Join, Leave and Merge Protocols whereas for our Join Protocol, only U_1 and U_{n+1} performs 2 exponentiations each while the rest need not perform any exponentiations. For our Merge Protocol, only U_1 and U_{n+1} perform 4 exponentiations each while all other users need not perform any exponentiations. Our Leave Protocol require only users $U_j \notin \mathcal{L}$ for j is odd to perform 3 exponentiations while $U_k \notin \mathcal{L}$ for k is even to perform 2 exponentiations. The proposed dynamic protocols are also more efficient than BD in terms of signature generation and verification.

6.5 Energy Analysis

In this section, we perform the total energy cost analysis of the proposed IKA, Join, Leave and Merge protocols. We assume that user devices are based on the 133MHz SA-1110 “StrongARM” microprocessor as described in Section 3.3.1 and these user devices can transmit and receive mes-

sages using two different communication transceivers, namely the 100kbps radio transceiver module and the IEEE 802.11 Spectrum24 WLAN card as discussed in Section 3.4.1. Next, we present Figure 6.1 that illustrates the total energy consumed by each node when running the proposed IKA protocol, three different variants of the BD protocol and the SSN scheme for different group sizes $n = 10, 50, 100$ and 500. Finally, we discuss and compare the energy costs of performing our proposed dynamic GKA protocols, namely Join, Leave and Merge Protocol, and re-executing authenticated BD protocol using ECDSA signature scheme. It should be noted that for the DSA signature scheme, a 172-bytes DSA certificate as described in Section 3.4.3 is used while for the ECDSA signature scheme, an 86-bytes ECDSA certificate as described in Section 3.4.3 is used.

Energy Cost Results for IKA Protocol

Figure 6.1: Energy Consumption Costs.

By considering the complexity in Table 6.1 and the energy costs in Tables 3.2 and 3.4, we obtained Figure 6.1 that shows the total energy costs of each node using the 133MHz “StrongARM” microprocessor with either 100kbps transceiver module or WLAN card. The total energy costs include the transmission and reception costs of all messages as well as the total computational costs for modular exponentiations, signature generation and signature verifications in each respective GKA protocol. For BD with ECDSA and DSA schemes, the total energy costs also include the transmission, reception and verification of all certificates used. In the figure, it can be seen that both the BD with ID-based Sakai et al’s SOK signature scheme using the 100kbps transceiver and

WLAN card (denoted as (e) and (f) in the figure) have the highest and second highest energy costs respectively due to the high cost of $(n - 1)$ MapToPoint operations and pairing operations in the signature verification of the SOK scheme. When using the WLAN card, our proposed IKA protocol (j) is the most efficient followed by the ID-based SSN scheme. Next, the BD with certificate-based ECDSA signature scheme (b) is slightly more efficient than the BD with certificate-based DSA signature scheme (d). When using the 100kbps transceiver module, our proposed IKA protocol (i) is again the most efficient followed by the ID-based SSN scheme, BD with certificate-based ECDSA signature scheme and BD with certificate-based DSA signature scheme. Hence, the results show that our proposed IKA protocol is the most energy-efficient when using either the 100kbps transceiver module or WLAN card.

Energy Cost Results for Dynamic Protocols

In this section, we present the energy cost analysis of our dynamic protocols and compare them with re-executing the authenticated BD protocol using the ECDSA signature scheme. We assume that each node in the group is using the 133MHz “StrongARM” microprocessor and the communication module used is the Spectrum24 WLAN card as presented in Tables 3.2 and 3.4 respectively. We also assume that the current group size $n = 100$ and the number of merging users $m = 20$. The energy cost results are shown in Table 6.3.

Table 6.3: Energy Cost for Dynamic Protocols

		Energy
BD Join	$U_1 - U_n$	$1.234J$
	U_{n+1}	$2.31J$
Proposed Join Protocol	U_1	$0.039J$
	U_n	$0.049J$
	U_{n+1}	$0.057J$
	Others	$1.34mJ$
BD Leave	Remain. Users	$1.179J$
Proposed Leave Protocol	$U_j, j = odd$	$0.160J$
	$U_k, k = even$	$0.150J$
BD Merge	Group $\mathcal{G}_A$ Users	$1.660J$
	Group $\mathcal{G}_B$ Users	$2.532J$
Proposed Merge Protocol	U_1	$0.079J$
	U_{n+1}	$0.079J$
	Others	$0.986mJ$

Based on the energy cost results shown in Table 6.3, it can be seen that the proposed dynamic protocols are much more suitable to be implemented in wireless networks where dynamic group membership events occurs frequently. If the BD protocol was to be implemented, then the battery life of each node will be depleted more rapidly due to the high energy costs. On the other hand, the proposed dynamic protocols consume much lower energy as shown in Table 6.3.

6.6 Summary

In this chapter, we presented an energy-efficient initial GKA protocol and three authenticated dynamic GKA protocols that take advantage of the GQ ID-based signature scheme. The complexity and energy cost analysis of the proposed protocols show that they are more suitable for wireless networks. We also presented the security analysis of our proposed protocols and showed that our proposed protocols are secure.

Chapter 7

Energy-Efficient and Scalable Group Key Agreement for Large Ad Hoc Networks

7.1 Introduction

An example of static¹ large-scale symmetric wireless ad hoc network is Wireless Sensor Networks (WSNs). WSNs are often used in military operations to detect and report events such as the movement of enemy troops, movement of enemy armored vehicles and detection of chemical occurrences occurring within range of the WSN [CKM00]. The sensor nodes are usually small, lightweight, inexpensive, low-power and battery-operated and they usually have no mobility. WSNs are more vulnerable to attacks and unauthorized access than traditional (wired) networks due to the characteristics of WSNs. For example, the sensor nodes are often deployed in hostile environments where they can be easily captured, compromised or manipulated by an adversary. Furthermore, the use of radio waves to transmit messages/data by sensor nodes enables an adversary to eavesdrop on the communication, inject false messages and replay or change old messages. As a result, key management and authentication are two essential security services to maintain proper operations of a WSN. As sensor nodes may collaborate with each other to more accurately detect a target and the reports/messages sent by a sensor node may be transmitted via several sensor nodes before reaching the remote command and control application, therefore a better security solution would be the use of a common group key that is known to all sensor nodes in the WSN to provide confidentiality and integrity for the messages/data sent. Solutions such as two-party key establishment protocols are

¹The term static means that the nodes in these networks are fixed and do not have mobility.

not desirable because they only provide end-to-end secure communications between two communicating sensor nodes and sensor nodes would require many different secret keys to communicate with different sensor nodes. Therefore, group key agreement (GKA) protocols provide better solutions for secure group communications in WSNs.

There is a need to employ energy-efficient GKA protocol so that the battery life of sensor nodes can be prolonged. Due to the large-scale nature of such applications, the GKA protocols also have to be scalable to handle large number of nodes in these networks. As mentioned in previous chapters, the Burmester-Desmedt (BD) [BD95] is an efficient and secure GKA protocol. However, as mentioned in [CNKW05], the BD requires each user to receive $\mathcal{O}(n)$ messages, therefore when the group size n is large, the BD becomes inefficient.

In this chapter, we propose an energy-efficient and scalable GKA protocol for static wireless ad hoc networks, especially large-scale symmetric wireless ad hoc networks, where *all* nodes are *low-power* energy constrained devices such as PDAs and sensor nodes. The proposed GKA protocol uses the C-H group model described in Section 2.12.1 of Chapter 2. A complexity analysis shows that our proposed protocol significantly reduces the number of messages to be received by each user in the BD protocol as well as the number of exponentiations and message reception by the last user in GDH. We then perform an analysis of the computational energy cost of the 133MHz SA-1110 “StrongARM” microprocessor and the communication energy costs of a radio transceiver module used in WSNs. The energy analysis shows that our proposed protocol is the most energy efficient as compared to the BD, GDH and TGDH protocol for different group sizes.

7.2 Group Key Agreement Protocol for Circular Hierarchical (C-H) Group Model

In this section, we presents the proposed Group Key Agreement (GKA) Protocol for the generalized Circular Hierarchical (C-H) Group Model described in Section 2.12.1 of Chapter 2. We have a large ad hoc network (which can be seen as a group) of n users that wishes to establish a group key for secure group communications. We can therefore establish a C-H group such that group size $n = c^h$. For more details of the C-H Group Model, we refer readers to Section 2.12.1 of Chapter 2.

During the initialization phase, two random large primes, q and p , are selected such that q divides $p - 1$. (p is 1024 bits and q is 160 bits.) Next, an element $g \in Z_p^*$ of order q is selected. Let H be a cryptographic hash function. The GKA protocol comprises of four phases. During the first three phases, the subgroup keys $K_{v,j}$ for each subgroup $\mathcal{G}_{v,j}$ are computed and the subgroup key at the highest layer $\mathcal{L}_0$ is the final group key K of the entire group. Finally in Phase 4, the final group key K is encrypted and broadcast down the layers to all subgroups securely using symmetric key cryptography.

Phase 1. The GKA protocol for the (C-H) group model starts at the lowest layer $\mathcal{L}_{h-1}$. Each user $U_i \in \mathcal{G}_{h-1,j}$ for $0 \leq i \leq c^h - 1$ and $j = \lfloor \frac{i}{c} \rfloor$ runs the BD protocol described in Section 2.7.6 with other members in its subgroup $\mathcal{G}_{h-1,j}$ and obtains subgroup key $K_{h-1,j}$.

Phase 2. This phase starts from second lowest layer $\mathcal{L}_{h-2}$ up to second highest layer $\mathcal{L}_1$. For each layer $\mathcal{L}_v$ for $1 \leq v \leq h - 2$ and subgroup $\mathcal{G}_{v,j}$ for $0 \leq j \leq c^v - 1$, each $U_i \in \mathcal{G}_{v,j}$ for $i = jc^{h-v}, jc^{h-v} + c^{h-v-1}, jc^{h-v} + 2c^{h-v-1}, \dots, jc^{h-v} + (c-1)c^{h-v-1}$ computes and broadcasts $z_{v,i} = g^{H(K_{v+1,k})} \bmod p$ in the first round of the BD protocol run in subgroup $\mathcal{G}_{v,j}$, where $k = \frac{i}{c^{h-v-1}}$ and $K_{v+1,k}$ is the subgroup key of subgroup $\mathcal{G}_{v+1,k}$ that $U_i \in \mathcal{G}_{v,j}$ controls at layer $\mathcal{L}_{v+1}$. Next, the second round and key computation round of the BD protocol are executed in subgroup $\mathcal{G}_{v,j}$ to obtain the subgroup key $K_{v,j}$.

This phase will complete when all subgroups at layer $\mathcal{L}_1$ have computed their respective subgroup keys $K_{1,j}$ for $0 \leq j \leq c - 1$.

Phase 3 - Key Computation. At the highest layer $\mathcal{L}_0$ and subgroup $\mathcal{G}_{0,0}$, each $U_i \in \mathcal{G}_{0,0}$ for $i = 0, c^{h-1}, 2c^{h-1}, \dots, (c-1)c^{h-1}$ computes and broadcasts $z_{0,i} = g^{H(K_{1,k})} \bmod p$ in the first round of the BD protocol run in subgroup $\mathcal{G}_{0,0}$, where $k = \frac{i}{c^{h-1}}$ and $K_{1,k}$ is the subgroup key of subgroup $\mathcal{G}_{1,k}$ that $U_i \in \mathcal{G}_{0,0}$ controls at Layer $\mathcal{L}_1$. After completing the BD protocol, each $U_i \in \mathcal{G}_{0,0}$ will be able to compute the subgroup key $K_{0,0}$ that is also the final group key K (for easier notations, we denote $K_{0,0}$ as K). Each $U_i \in \mathcal{G}_{0,0}$ then uses the subgroup key $K_{1,k}$ of its subgroup $\mathcal{G}_{1,k}$ and a symmetric key encryption scheme SKE to produce and broadcast $\mathcal{E}_{K_{1,k}}(K)$ to its respective subgroup $\mathcal{G}_{1,k}$.

Phase 4 - Key Distribution

(i) This phase starts from second highest layer $\mathcal{L}_1$ down to second lowest layer $\mathcal{L}_{h-2}$. For each

layer $\mathcal{L}_v$ for $1 \leq v \leq h-2$ and subgroup $\mathcal{G}_{v,j}$ for $0 \leq j \leq c^v - 1$, each $U_i \in \mathcal{G}_{v,j}$ for $i = jc^{h-v}, jc^{h-v} + c^{h-v-1}, jc^{h-v} + 2c^{h-v-1}, \dots, jc^{h-v} + (c-1)c^{h-v-1}$ first decrypts the encrypted message $\mathcal{E}_{K_{v,j}}(K)$ received from its subgroup controller who belongs to a subgroup in the previous layer $\mathcal{L}_{v-1}$ and obtains the final group key K . Next, each $U_i \in \mathcal{G}_{v,j}$ uses the subgroup key $K_{v+1,k}$ where $k = \frac{i}{c^{h-v-1}}$ and a symmetric key encryption scheme SKE to produce and broadcast $\mathcal{E}_{K_{v+1,k}}(K)$ to its subgroup $\mathcal{G}_{v+1,k}$ at lower layer $\mathcal{L}_{v+1}$.

(ii) At lowest layer $\mathcal{L}_{h-1}$, each $U_i \in \mathcal{G}_{h-1,j}$ for $0 \leq i \leq c^h - 1$ and $j = \lfloor \frac{i}{c} \rfloor$ decrypts $\mathcal{E}_{K_{h-1,j}}(K)$ and obtains the final group key K .

Here, we assume that all messages sent are digitally signed with a sufficiently strong signature scheme and all messages received are verified to ensure authentication in the group key agreement.

7.2.1 Security Analysis

In this section, we provide the security analysis of the proposed protocol as follows:

Theorem 24 *An adversary $\mathcal{A}$ is not able to obtain the subgroup keys $K_{v,j}$ for $1 \leq v \leq h-1$ and $0 \leq j \leq c^v - 1$ and the final group key K in the proposed GKA protocol if the BD protocol and the symmetric key encryption scheme SKE are secure.*

Proof: In this proof, we consider two cases, that is, (a) the adversary $\mathcal{A}$ tries to obtain the subgroup key $K_{v,j}$ for $1 \leq v \leq h-1$ and $0 \leq j \leq c^v - 1$ and the final group key K in the up flow stage (Phase 1, 2 and 3) and (b) the adversary $\mathcal{A}$ tries to obtain the subgroup key $K_{v,j}$ for $1 \leq v \leq h-1$ and $0 \leq j \leq c^v - 1$ and the final group key K in the down flow stage (Phase 4). For case (a), first consider Phase 1, as the subgroup keys $K_{h-1,j}$ of $\mathcal{G}_{h-1,j}$ for $0 \leq j \leq c^{h-1} - 1$ are generated using the BD protocol, hence the adversary $\mathcal{A}$ is not able to obtain $K_{h-1,j}$ as the BD protocol is assumed to be secure. In Phase 2 and 3, for layers $\mathcal{L}_v$ for $0 \leq v \leq h-2$ and subgroup $\mathcal{G}_{v,j}$ for $0 \leq j \leq c^v - 1$ where $\mathcal{G}_{0,0}$ is the subgroup at the highest layer $\mathcal{L}_0$ (i.e. $v = 0$), each $U_i \in \mathcal{G}_{v,j}$ for $i = jc^{h-v}, jc^{h-v} + c^{h-v-1}, jc^{h-v} + 2c^{h-v-1}, \dots, jc^{h-v} + (c-1)c^{h-v-1}$ computes and broadcasts $z_{v,i} = g^{H(K_{v+1,k})} \bmod p$ in the first round of the BD protocol, where $K_{v+1,k}$ is the subgroup key also generated by the BD protocol and $k = \frac{i}{c^{h-v-1}}$. Next, the second round and key computation round of the BD protocol are executed to obtain the subgroup key $K_{v,j}$ where $K_{0,0}$

is the final group key K . As BD protocol is provably secure, hence the adversary $\mathcal{A}$ is not able to obtain the subgroup keys $K_{v,j}$ for $0 \leq v \leq h-2$ and $0 \leq j \leq c^v-1$ (where $K_{0,0}$ is the final group key K) in Phase 2 and 3. Therefore the adversary $\mathcal{A}$ is not able to obtain the subgroup keys $K_{v,j}$ for $1 \leq v \leq h-1$ and $0 \leq j \leq c^v-1$ and final group key K in the up flow stage (Phase 1, 2 and 3). Now we consider case (b) as follows: As the final group key K is the plaintext of $\mathcal{E}_{K_{v,j}}(K)$ for $1 \leq v \leq h-1$ and $0 \leq j \leq c^v-1$ in the down flow stage (Phase 4). Therefore, it is not possible for the adversary $\mathcal{A}$ to recover the final group key K from $\mathcal{E}_{K_{v,j}}(K)$ as it is assumed that the symmetric key encryption scheme SKE is secure. Therefore, we conclude that the adversary $\mathcal{A}$ is not able to obtain the subgroup keys $K_{v,j}$ for $1 \leq v \leq h-1$ and $0 \leq j \leq c^v-1$ and the final group key K from the proposed protocol. ■

7.3 Complexity Analysis

We compare the computational and communication complexity of our protocol with three other protocols, namely the BD [BD95], TGDH [KPT04b] and GDH (based on GDH.3 [STW00]). In Table 7.1, the computational cost refers to the number of modular exponentiations required to compute the final group key while the communication cost is represented by the number of messages transmitted and received. As mentioned in [CKM00], [HV02] and [TWL05], the computational cost for modular exponentiations are several orders of magnitude higher than symmetric key cryptography, therefore we assume that the computational complexity of symmetric key encryption/decryption to be negligible as compared to modular exponentiations.

The notations c , h and n in Table 7.1 refers to the number of members in each subgroup, the number of layers (or height of the tree for TGDH) and group size respectively. It should be noted that the height of the tree in TGDH is $h = \log_2 n$ while the number of layers or height of the (C-H) group model in our proposed protocol is $h = \log_c n$. The notations $\mathcal{G}_{0,0}$, $\mathcal{G}_{v,j}$ ($1 \leq v \leq h-3$ and $0 \leq j \leq c^v-1$) and $\mathcal{G}_{h-2,j}$ ($0 \leq j \leq c^{h-2}-1$) denotes the subgroup members in $\mathcal{G}_{0,0}$, $\mathcal{G}_{v,j}$ ($1 \leq v \leq h-3$ and $0 \leq j \leq c^v-1$) and $\mathcal{G}_{h-2,j}$ ($0 \leq j \leq c^{h-2}-1$) respectively. As mentioned earlier, the number of users in each subgroup at every layer for our proposed protocol need not be fixed and can vary in different subgroups depending on the application requirements. To illustrate our proposed protocol and without loss of generality, we assume that the network is balanced with

Table 7.1: Computation and Communication Cost

		Exp.	Msg Sent	Msg Received
BD		3	2	$n + 1$
TGDH		$2h$	h	h
GDH	$U_1 - U_{n-2}$	3	2	3
	U_{n-1}	2	1	2
	U_n	n	1	$n - 1$
Proposed Protocol	$\mathcal{G}_{0,0}$	$3h$	$3h - 1$	$h(c + 1)$
	$\mathcal{G}_{v,j}$ for $1 \leq v \leq h - 3$	$3(h - v)$	$2(h - v) + (h - (v + 1))$	$((h - v)(c + 1)) + 1$
	$\mathcal{G}_{h-2,j}$	6	5	$2c + 3$
	Users	3	2	$c + 2$

every subgroup containing the same number of users, c .

As shown in Table 7.1, the BD requires each user to perform three modular exponentiations and broadcast two messages. However, each user has to receive $n + 1$ messages, which will be large if the group size n is large. For TGDH, each user has to compute $2h$ modular exponentiations, broadcasts and receives h messages respectively. The GDH protocol requires users U_i for $1 \leq i \leq n - 2$ to compute three exponentiations, transmit two messages and receive three messages in total. However, the last message received by U_i is a large message broadcast by U_n that contains $n - 1$ keying materials for all users in the group. Although second last user U_{n-1} only has to compute two modular exponentiations, broadcast one message and receive two messages, however, like U_i , the last message received by U_{n-1} is a large message containing $n - 1$ keying materials from U_n . The last user U_n in GDH has the highest amount of computational and communication costs. Although U_n broadcast only one message, this message is a large message containing $n - 1$ 1024-bit keying materials for all other users to compute the group key.

Our proposed protocol reduces the number of messages received by all users as compared to the BD. Furthermore, our protocol requires fewer modular exponentiations and messages to be transmitted than any user in TGDH. This is because the height of the tree in TGDH is defined as $h = \log_2 n$ whereas the total number of layers or height of the (C-H) group model in our proposed protocol is defined as $h = \log_c n$. For example, for a group size of $n = 512$, the height of the TGDH tree is $h = \log_2 512 = 9$ while the number of layers or height of the (C-H) group model in our

proposed protocol is $h = \log_8 512 = 3$, where $c = 8$. This advantage of our protocol over TGDH can be clearly seen in the next section (Energy Analysis). Similarly, our proposed protocol also logarithmically scales down the number of exponentiations and received messages required by the last user U_n in GDH. Although the number of received messages in our proposed protocol depends on the subgroup size c , however, it should be noted that in GDH, the last message received by each user U_i for $1 \leq i \leq n - 1$ is a large message containing $n - 1$ 1024-bit keying materials.

7.4 Energy Analysis

In this section, we perform the computational (modular exponentiation) energy cost analysis of the 133MHz SA-1110 “StrongARM” microprocessor typically used by devices in ad hoc networks such as PDAs and sensor nodes and the communication (transmission and reception of messages) energy costs analysis of a radio transceiver module used in the Rockwell Scientific WINS sensor node. Based on the complexity analysis in Table 7.1 and the energy costs of the Rockwell Scientific WINS sensor node, which uses the 133MHz SA-1110 “StrongARM” microprocessor described in Section 3.3.1 and the 100kbps transceiver module described in Section 3.4.1, we obtain the energy required by all users to establish a group key using the BD, GDH, TGDH and our proposed protocol for different group size n . The computational, transmission, reception of messages and total (both computational and communication) energy costs comparison of our proposed protocol with the BD, GDH and TGDH protocols are presented in Figures 7.1, 7.2, 7.3 and 7.4 respectively. The Y-axis in these figures measures the energy consumed in milliJoules (mJ) while the X-axis shows the group size ($n = 8, 64, 4096$ and 32768). Figure 7.5 shows the total energy consumed by the BD, GDH, TGDH and our proposed protocol. The total energy consumption by each GKA protocol was obtained by adding up the total energy consumed by every node in that particular GKA protocol. The Y-axis in Figure 7.5 shows the total energy consumed by all nodes in Joules (J) while the X-axis shows the group size n . The notations $\mathcal{G}_{0,0}$ and $\mathcal{G}_{v,j}$ ($1 \leq v \leq 3$ and $0 \leq j \leq c^v - 1$) denotes the subgroup members in $\mathcal{G}_{0,0}$ and $\mathcal{G}_{v,j}$ ($1 \leq v \leq 3$ and $0 \leq j \leq c^v - 1$) respectively.

For our proposed protocol, the number of users in each subgroup at every layer need not be fixed and can vary in different subgroups depending on the application requirements. For illustration purposes in our energy analysis, we will assume that the network is balanced with every subgroup

containing the same number of users, $c = 8$. For example, for a group size of $n = 4096$, we are using a (C-H) group model for GKA where $c = 8$ and $h = 4$ such that $4096 = 8^4$. It should be noted that in all the figures, we denote the subgroup members in the lowest layer $\mathcal{L}_{h-1}$ of our proposed protocol as **Users**.

Figure 7.1: Computational Energy Consumption Costs.

Figure 7.2: Transmission (Communication) Energy Consumption Costs.

Figure 7.3: Reception (Communication) Energy Consumption Costs.

Figure 7.4: Total (Computational and Communication) Energy Consumption Costs.

Computational Energy Cost

Figure 7.1 shows the computational energy cost comparison. Although the second last user U_{n-1} in GDH consumes the least amount of energy for different group size, however the last user U_n in

GDH consumes the most energy. In fact, U_n in GDH consumes $478.41J$ for exponentiation when the group size $n = 32768$, this suggest that U_n has to be a high-power node such as a base station or access point when the group size is large. In comparison, each member in subgroup $\mathcal{G}_{0,0}$ in our protocol, which consumes the most amount of energy for exponentiation, consumes only $0.219J$ for group size $n = 32768$. Figure 7.1 also shows that all nodes in our proposed protocol consumes less energy than the users in TGDH regardless of the group size n . From more detailed examination, each user in TGDH consumes almost double the energy required by members in subgroup $\mathcal{G}_{0,0}$ in our protocol. Besides having lower energy costs than users in TGDH, the users (subgroup members at the lowest layer $\mathcal{L}_{h-1}$) in our proposed protocol, like all users in the BD protocol and users U_i in GDH, have the second lowest energy cost ($43.8mJ$) regardless of the group size n .

Transmission (Communication) Energy Cost

The transmission energy costs are shown in Figure 7.2. User U_{n-1} of GDH consumes the least energy ($11.1mJ$) while the users in our proposed protocol, users U_i in GDH and all users in BD have the second lowest energy costs ($22.2mJ$) regardless of group size n . Figure 7.2 also shows that all nodes in our proposed protocol have lower energy costs than the users in TGDH. The last user U_n of GDH have the highest energy cost for transmission because the message size to be broadcast by U_n increases linearly as group size n increases.

Reception (Communication) Energy Cost

Figure 7.3 shows the energy cost of reception by all users. From the figure, it can be seen that all users (U_1 to U_n) in GDH and BD consumes the most energy because users U_1 to U_{n-1} in GDH has to receive a large message from user U_n to compute the group key and user U_n in GDH has to receive messages from all other users before it can compute the group key and keying materials for other users. For users in BD, the reason for large energy consumption is because each user has to receive messages from all other nodes before it can compute the final group key. Figure 7.3 also shows that all nodes in our proposed protocol have significantly lower energy cost for reception as compared to all users in BD and GDH. This is especially true when the group size n is large. TGDH is more efficient, but when the group size n is above 512 (when the group size becomes large), the users

(subgroup members of the lowest layer $\mathcal{L}_{h-1}$) in our protocol consume lower amount of energy than the users in TGDH. This is because in TGDH, as the group size n becomes larger, the height of the tree h also increases and therefore, the number of messages received by each user node in TGDH also increases linearly with h as shown in Table 7.1. On the other hand, the number of messages received by user nodes in our protocol is only dependent on c (the number of subgroup members which is a constant) and independent of the group size n and total number of layers h .

Total Energy Cost

The total (both computational and communication) energy cost for GKA is presented in Figure 7.4. It can be seen that our proposed protocol shows significant improvement over the original BD protocol. This is primarily because the BD protocol requires each node to receive messages from all other nodes before computing the group key whereas in our proposed protocol, the number of messages to be received by each node is reduced by placing each node in the (C-H) group model as shown in Figure 2.11, which has smaller subgroup size (each subgroup has c members only) in different layers. Our proposed protocol is also more efficient than TGDH since all other nodes, except the members in subgroup $\mathcal{G}_{0,0}$, have *lower* energy costs. Furthermore, as shown in Figure 7.4, the members in subgroup $\mathcal{G}_{0,0}$ have only slightly higher energy costs than all users in TGDH. Besides, there are only $c = 8$ members in subgroup $\mathcal{G}_{0,0}$ in our proposed protocol as compared to all users in TGDH. Users U_1 to U_{n-1} in GDH, like all users in BD, consume the second highest amount of energy while U_n in GDH consumes the most amount of energy. The main reason for the large energy consumption for users U_1 to U_{n-1} in GDH is due to the reception costs as mentioned earlier while user U_n in GDH not only has to receive $n - 1$ messages, but also perform n modular exponentiations and broadcast a large message.

Total Energy Consumed by each GKA protocol

Figure 7.5 shows the total energy consumption by each of the four GKA protocols, namely BD, TGDH, GDH and our proposed protocol. The total energy consumption in Figure 7.5 were obtained by adding up the total energy consumed by every node in each respective GKA protocol.

From Figure 7.5, we can see that our proposed protocol is the most efficient. The BD and GDH

Figure 7.5: Total Energy Consumption by each GKA protocol.

consume the most amount of energy. For the BD, this large energy consumption is due to the large amount of messages to be received by each node to compute the group key K whereas for the GDH, it is mainly due to the large message containing $n - 1$ 1024-bit keying materials to be received by each user U_1 to U_{n-1} to compute the group key. Figure 7.5 also shows that our proposed protocol is more energy efficient than TGDH as the total energy consumed by our protocol is much lower than that of TGDH. Based on the results shown in Figure 7.4 and Figure 7.5, our proposed protocol is the most energy-efficient and scalable GKA protocol for large-scale static ad hoc networks.

7.5 Summary

In this chapter, we proposed an energy-efficient and scalable GKA protocol for large-scale static ad hoc networks, which uses a generalized circular hierarchical (C-H) group model. A complexity analysis and energy costs analysis indicate that our proposed protocol is the most energy-efficient and scalable GKA protocol as compared to three other GKA protocols.

Chapter 8

Low-Power Group Key Agreement for Heterogeneous Wireless Networks

8.1 Introduction

Unlike Chapter 7, which discussed on static symmetric wireless ad hoc networks where all nodes have equal computational power, we focus on static asymmetric Heterogeneous Wireless Networks (HWNs) which include Wireless Mesh Networks (WMNs) [AWW05] and certain types of ad hoc networks in this chapter. In asymmetric HWNs, there are nodes that are high power devices without energy constraints but user nodes in these networks are typically low power energy constrained devices. Hence, the GKA protocol for HWNs also has to be energy-efficient to prolong the battery lifespan of the user nodes. Furthermore, in large scale HWNs, there could be thousands of user nodes, hence the GKA protocol also has to be scalable. The high power nodes in these networks could serve as trusted controllers to handle the bulk of computation to reduce the burden of low power user nodes.

The NLKW [NLKW05] and BD [BD95] are two efficient GKA protocols for user nodes in terms of exponentiations and message transmissions. However, each user in BD has to receive $\mathcal{O}(n)$ messages and each user in NLKW has to receive a large message containing $\mathcal{O}(n)$ keying materials for other users to compute the group key, where n is the group size. To improve efficiency and scalability, one technique is to design hierarchical GKA protocols such as Nam et al's Hierarchical GKA Scheme (NHS) [NKW05] and the proposed protocol described in Chapter 7 (BD-BD).

In this chapter, we present an energy efficient and scalable authenticated GKA protocol, which uses our proposed Contributory Ring-Centralized (ContRi-Central) group model (described in Section 2.12.2 of Chapter 2), for static asymmetric HWNs. A complexity analysis of our proposed protocol with the two hierarchical GKA protocols (NHS and BD-BD) and two other GKA protocols (BD and NLKW) shows that our proposed protocol is comparatively efficient. We also provided an energy analysis that includes the computational energy costs of the 133MHz SA-1110 “StrongARM” microprocessor and the communication energy costs of the Spectrum24 WLAN card while running the five different GKA protocols. The energy analysis shows that users nodes in our proposed protocol are the most energy-efficient, consuming the least energy for GKA.

8.2 GKA Protocol for ContRi-Central Group Model

We present our authenticated GKA protocol for the multi-layer ContRi-Central Group Model described in Section 2.12.2 of Chapter 2. We have a large heterogeneous wireless network of n users that wishes to establish a group key for secure group communications. The GKA protocol, which comprises of five phases, will be used to establish the various subgroup keys $K_{v,j}$ at each layer $\mathcal{L}_v$ for $1 \leq v \leq h-1$ and $0 \leq j \leq c^v-1$ and the subgroup key $K_{0,0}$ of the control group CG is the final group key K (for easier notations, we denote $K_{0,0}$ as K). We refer readers to Section 2.12.2 in Chapter 2 for more details of the ContRi-Central Group Model.

During the initialization phase, two random large primes, q and p , are selected such that q divides $p-1$. (p is 1024 bits and q is 160 bits.) Next, an element $g \in Z_p^*$ of order q is selected. Let H be a cryptographic hash function. We assume that the DSA signature scheme [DSA94] is used in the proposed protocol. Each subgroup controller $S_{v,j}$ for $0 \leq v \leq h-2$ and $0 \leq j \leq c^{v+1}-1$ and users $S_{h-1,i}$ for $0 \leq i \leq c^h-1$ will compute their respective DSA private and public key pair using the key generation phase of the DSA scheme shown in Section 2.3.4, and register and obtain the DSA certificate $\text{cert}[S_{v,j}]$ and $\text{cert}[S_{h-1,i}]$ respectively from the Certification Authority. In the first four phases, the subgroup keys $K_{v,j}$ for $1 \leq v \leq h-1$ and $0 \leq j \leq c^v-1$ of each subgroup $\mathcal{G}_{v,j}$ and the final group key K , which is the subgroup key $K_{0,0}$ of the control group CG at $\mathcal{L}_0$, are computed. In Phase 5, the final group key K is encrypted and broadcast down the various layers to all subgroups securely using symmetric key cryptography.

The details of the GKA protocol are as follows:

Phase 1 : The GKA protocol will start at the lowest layer $\mathcal{L}_{h-1}$. Each user $\mathbf{S}_{h-1,j}$ for $0 \leq j \leq c^h - 1$ selects random $r_{h-1,j} \in Z_q^*$ and computes $z_{h-1,j} = g^{r_{h-1,j}} \bmod p$. $\mathbf{S}_{h-1,j}$ then signs $\mathbf{S}_{h-1,j}||z_{h-1,j}$ and obtains signature $\sigma_{h-1,j}$ and sends $\{\mathbf{S}_{h-1,j}||z_{h-1,j}||\sigma_{h-1,j}||\text{cert}[\mathbf{S}_{h-1,j}]\}$ to its subgroup controller $\mathbf{S}_{h-2,i}$ at layer $\mathcal{L}_{h-2}$ where $i = \lfloor \frac{j}{c} \rfloor$.

Phase 2a :

(i) This phase starts from second lowest layer $\mathcal{L}_{h-2}$ up to second highest layer $\mathcal{L}_1$. For each layer $\mathcal{L}_v$ for $1 \leq v \leq h-2$, each subgroup controller $\mathbf{S}_{v,j}$ for $0 \leq j \leq c^{v+1} - 1$ of subgroup $\mathcal{G}_{v+1,j} = \{\mathbf{S}_{v+1,jc}, \mathbf{S}_{v+1,jc+1}, \mathbf{S}_{v+1,jc+2}, \dots, \mathbf{S}_{v+1,jc+c-1}\}$ first verifies the certificates $\text{cert}[\mathbf{S}_{v+1,jc+k}]$ and signatures $\sigma_{v+1,jc+k}$ sent by its subgroup members $\mathbf{S}_{v+1,jc+k} \in \mathcal{G}_{v+1,j}$ for $0 \leq k \leq c-1$. If the verifications are not correct, then each subgroup controller $\mathbf{S}_{v,j}$ rejects $\mathbf{S}_{v+1,jc+k}$. Otherwise, $\mathbf{S}_{v,j}$ selects random $r_{v,j} \in Z_q^*$ and computes $\bar{z}_{v,j} = g^{r_{v,j}} \bmod p$ and Diffie-Hellman (DH) keys $T_{v+1,jc+k} = z_{v+1,jc+k}^{r_{v,j}} \bmod p$ that it shares with each subgroup member $\mathbf{S}_{v+1,jc+k}$. Each $\mathbf{S}_{v,j}$ then computes the subgroup key $K_{v+1,j} = \bar{z}_{v,j}^{r_{v,j}} \cdot \prod_{k=0}^{c-1} T_{v+1,jc+k} \bmod p$. Next, $\mathbf{S}_{v,j}$ encrypts $K_{v+1,j}||\mathbf{S}_{v,j}$ using $T_{v+1,jc+k}$ and a symmetric key encryption SKE to obtain $\mathbf{c}_{v+1,jc+k} = \mathcal{E}_{T_{v+1,jc+k}}(K_{v+1,j}||\mathbf{S}_{v,j})$ for $0 \leq k \leq c-1$. Each $\mathbf{S}_{v,j}$ then computes $z_{v,j} = g^{H(K_{v+1,j})} \bmod p$. Next, each $\mathbf{S}_{v,j}$ signs $\mathbf{S}_{v,j}||z_{v,j}$ and $\mathbf{S}_{v,j}||\bar{z}_{v,j}||\mathbf{c}_{v+1,jc+k}$ for $0 \leq k \leq c-1$ and obtains signatures $\sigma_{v,j}$ and $\bar{\sigma}_{v,j}$ respectively. Finally, $\mathbf{S}_{v,j}$ broadcasts $\{\mathbf{S}_{v,j}||z_{v,j}||\sigma_{v,j}||\text{cert}[\mathbf{S}_{v,j}]\}$ to its subgroup controller $\mathbf{S}_{v-1,i}$ at layer $\mathcal{L}_{v-1}$ where $i = \lfloor \frac{j}{c} \rfloor$ and $\{\mathbf{S}_{v,j}||\bar{z}_{v,j}||\mathbf{c}_{v+1,jc+k}||\bar{\sigma}_{v,j}||\text{cert}[\mathbf{S}_{v,j}]\}$ for $0 \leq k \leq c-1$ to its subgroup $\mathcal{G}_{v+1,j}$ at layer $\mathcal{L}_{v+1}$.

(ii) For Layer $\mathcal{L}_0$, each $\mathbf{S}_{0,j}$ for $0 \leq j \leq c-1$ of subgroup $\mathcal{G}_{1,j} = \{\mathbf{S}_{1,jc}, \mathbf{S}_{1,jc+1}, \mathbf{S}_{1,jc+2}, \dots, \mathbf{S}_{1,jc+c-1}\}$ first verifies the certificates $\text{cert}[\mathbf{S}_{1,jc+k}]$ and signatures $\sigma_{1,jc+k}$ sent by its subgroup members $\mathbf{S}_{1,jc+k} \in \mathcal{G}_{1,j}$ for $0 \leq k \leq c-1$. If the verifications are not correct, then each $\mathbf{S}_{0,j}$ rejects $\mathbf{S}_{1,jc+k}$. Otherwise, $\mathbf{S}_{0,j}$ selects random $r_{0,j} \in Z_q^*$ and computes $\bar{z}_{0,j} = g^{r_{0,j}} \bmod p$ and DH keys $T_{1,jc+k} = z_{1,jc+k}^{r_{0,j}} \bmod p$ that it shares with each subgroup member $\mathbf{S}_{1,jc+k}$. Each $\mathbf{S}_{0,j}$ then computes the subgroup key $K_{1,j} = \bar{z}_{0,j}^{r_{0,j}} \cdot \prod_{k=0}^{c-1} T_{1,jc+k} \bmod p$. Next, $\mathbf{S}_{0,j}$ encrypts $K_{1,j}||\mathbf{S}_{0,j}$ using $T_{1,jc+k}$ and a symmetric key encryption SKE and obtains $\mathbf{c}_{1,jc+k} = \mathcal{E}_{T_{1,jc+k}}(K_{1,j}||\mathbf{S}_{0,j})$ for $0 \leq k \leq c-1$. Each $\mathbf{S}_{0,j}$ then computes $z_{0,j} = g^{H(K_{1,j})} \bmod p$. Next, each $\mathbf{S}_{0,j}$ signs $\mathbf{S}_{0,j}||z_{0,j}$ and $\mathbf{S}_{0,j}||\bar{z}_{0,j}||\mathbf{c}_{1,jc+k}$ for $0 \leq k \leq c-1$ to obtain $\sigma_{0,j}$ and $\bar{\sigma}_{0,j}$ respectively. $\mathbf{S}_{0,j}$ broadcasts $\{\mathbf{S}_{0,j}||z_{0,j}||\sigma_{0,j}||\text{cert}[\mathbf{S}_{0,j}]\}$ to CG

at Layer $\mathcal{L}_0$ and $\{\mathbf{S}_{0,j}||\bar{z}_{0,j}||\mathbf{c}_{1,jc+k}||\bar{\sigma}_{0,j}||\mathbf{cert}[\mathbf{S}_{0,j}]\}$ for $0 \leq k \leq c-1$ to its subgroup $\mathcal{G}_{1,j}$ at layer $\mathcal{L}_1$.

Phase 2b : This phase starts from second highest layer $\mathcal{L}_1$ down to lowest layer $\mathcal{L}_{h-1}$. For each layer $\mathcal{L}_v$ for $1 \leq v \leq h-1$, each subgroup member $\mathbf{S}_{v,j}$ for $0 \leq j \leq c^{v+1}-1$ of $\mathcal{G}_{v,i}$ where $i = \lfloor \frac{j}{c} \rfloor$ first verifies the certificate $\mathbf{cert}[\mathbf{S}_{v-1,i}]$ and signature $\bar{\sigma}_{v-1,i}$ from its subgroup controller $\mathbf{S}_{v-1,i}$. If the verifications are not correct, then $\mathbf{S}_{v,j}$ rejects $\mathbf{S}_{v-1,i}$. Otherwise, $\mathbf{S}_{v,j}$ computes the DH key $T_{v,j}$ that it shares with its subgroup controller $\mathbf{S}_{v-1,i}$ as follows:

$$T_{v,j} = \begin{cases} \bar{z}_{v-1,i}^{\mathbf{H}(K_{v+1,j})} \bmod p & \text{for } 1 \leq v \leq h-2, \\ \bar{z}_{v-1,i}^{r_{v,j}} \bmod p & \text{for } v = h-1. \end{cases}$$

Next, each $\mathbf{S}_{v,j}$ uses $T_{v,j}$ to decrypt $\mathbf{c}_{v,j}$ and obtains the subgroup key $K_{v,i}$ and identity $\mathbf{S}_{v-1,i}$. If the identity $\mathbf{S}_{v-1,i}$ is not decrypted correctly, then $\mathbf{S}_{v,j}$ rejects $\mathbf{c}_{v,j}$. Otherwise, $\mathbf{S}_{v,j}$ accepts subgroup key $K_{v,i}$.

Phase 3 : For the Control Group (CG) at Layer $\mathcal{L}_0$, each subgroup controller $\mathbf{S}_{0,j}$ for $0 \leq j \leq c-1$ first verifies the certificates $\mathbf{cert}[\mathbf{S}_{0,k}]$ for $0 \leq k \neq j \leq c-1$. If the verifications are not correct, then each $\mathbf{S}_{0,j}$ rejects $\mathbf{S}_{0,k}$. Otherwise, $\mathbf{S}_{0,j}$ verifies signatures $\sigma_{0,j-1}$ and $\sigma_{0,j+1}$. If the verifications are not correct, then each $\mathbf{S}_{0,j}$ rejects $\mathbf{S}_{0,j-1}$ and $\mathbf{S}_{0,j+1}$. Otherwise, $\mathbf{S}_{0,j}$ computes

$$\begin{aligned} X_{0,j} &= \left(\frac{z_{0,j+1}}{z_{0,j-1}} \right)^{\mathbf{H}(K_{1,j})} \\ &= g^{\mathbf{H}(K_{1,j})\mathbf{H}(K_{1,j+1}) - \mathbf{H}(K_{1,j-1})\mathbf{H}(K_{1,j})} \bmod p \end{aligned} \quad (8.1)$$

where $z_{0,-1} = z_{0,c-1}$, $z_{0,c} = z_{0,0}$, $\mathbf{H}(K_{1,-1}) = \mathbf{H}(K_{1,c-1})$ and $\mathbf{H}(K_{1,c}) = \mathbf{H}(K_{1,0})$. Next, each $\mathbf{S}_{0,j}$ signs $\mathbf{S}_{0,j}||X_{0,j}$ to obtain signature $\sigma'_{0,j}$ and broadcasts $\{\mathbf{S}_{0,j}||X_{0,j}||\sigma'_{0,j}\}$.

Phase 4 : For the Control Group (CG) at Layer $\mathcal{L}_0$, each subgroup controller $\mathbf{S}_{0,j}$ for $0 \leq j \leq c-1$ first verifies the signatures $\sigma'_{0,k}$ for $0 \leq k \neq j \leq c-1$ from the previous phase. If the verifications are not correct, then each $\mathbf{S}_{0,j}$ rejects $\mathbf{S}_{0,k}$. Otherwise, $\mathbf{S}_{0,j}$ computes the final group key K which is also the control group (CG) key $K_{0,0}$ as follows:

$$\begin{aligned} K &= (z_{0,j-1}^{\mathbf{H}(K_{1,j})})^c \cdot X_{0,j}^{c-1} \cdot X_{0,j+1}^{c-2} \cdots X_{0,j-2} \\ &= g^{\mathbf{H}(K_{1,0})\mathbf{H}(K_{1,1}) + \mathbf{H}(K_{1,1})\mathbf{H}(K_{1,2}) + \dots + \mathbf{H}(K_{1,c-1})\mathbf{H}(K_{1,0})} \bmod p \end{aligned} \quad (8.2)$$

where $z_{0,-1} = z_{0,c-1}$, $X_{0,-2} = X_{0,c-2}$ and $X_{0,c} = X_{0,0}$. Each $S_{0,j}$ then encrypts $K||S_{0,j}$ using its subgroup key $K_{1,j}$ and symmetric key encryption SKE to obtain $c'_{0,j} = \mathcal{E}_{K_{1,j}}(K||S_{0,j})$. Finally, each $S_{0,j}$ sends $\{S_{0,j}||c'_{0,j}\}$ to its subgroup $\mathcal{G}_{1,j}$ at layer $\mathcal{L}_1$.

Phase 5: Key Distribution :

- (i) This phase starts from second highest layer $\mathcal{L}_1$ down to second lowest layer $\mathcal{L}_{h-2}$. For each layer $\mathcal{L}_v$ for $1 \leq v \leq h-2$, each subgroup member $S_{v,j}$ for $0 \leq j \leq c^{v+1} - 1$ of $\mathcal{G}_{v,i}$ where $i = \lfloor \frac{j}{c} \rfloor$ first decrypts $c'_{v-1,i} = \mathcal{E}_{K_{v,i}}(K||S_{v-1,i})$ from its subgroup controller $S_{v-1,i}$ and obtains K and identity $S_{v-1,i}$. If the identity $S_{v-1,i}$ was not decrypted correctly, then $S_{v,j}$ rejects $c'_{v-1,i}$. Otherwise, $S_{v,j}$ accepts the group key K . Next, $S_{v,j}$ encrypts $K||S_{v,j}$ using the subgroup key $K_{v+1,j}$ of subgroup $\mathcal{G}_{v+1,j}$ that it controls at layer $\mathcal{L}_{v+1}$ and symmetric key encryption SKE and obtains $c'_{v,j} = \mathcal{E}_{K_{v+1,j}}(K||S_{v,j})$. $S_{v,j}$ then broadcast $\{S_{v,j}||c'_{v,j}\}$ to its subgroup $\mathcal{G}_{v+1,j}$ at layer $\mathcal{L}_{v+1}$.
- (ii) At lowest layer $\mathcal{L}_{h-1}$, each $S_{h-1,j}$ for $0 \leq j \leq c^h - 1$ who is a subgroup member of $\mathcal{G}_{h-1,i}$ where $i = \lfloor \frac{j}{c} \rfloor$ first decrypts $c'_{h-2,i} = \mathcal{E}_{K_{h-1,i}}(K||S_{h-2,i})$ from its subgroup controller $S_{h-2,i}$. If the identity $S_{h-2,i}$ was not decrypted correctly, then $S_{h-1,j}$ rejects $c'_{h-2,i}$. Otherwise, $S_{h-1,j}$ accepts the group key K .

8.3 Security Analysis

In this section, we analyze the security of our proposed protocol. We consider two cases, that is, the up flow information (Phase 1, 2a, 3 and 4) and down flow information (Phase 2b and 5). The up flow information is the information sent from the lowest layer $\mathcal{L}_{h-1}$ up to the highest layer $\mathcal{L}_0$ to form the subgroup keys $K_{v,j}$ for $1 \leq v \leq h-1$ and $0 \leq j \leq c^v - 1$ and final group key K which is the subgroup key $K_{0,0}$ of CG (for easier notations, we denote $K_{0,0}$ as K). The down flow information is basically a dissemination of the subgroup keys $K_{v,j}$ to the respective subgroups $\mathcal{G}_{v,j}$ in Phase 2b and final group key K from the highest layer $\mathcal{L}_0$ down to lowest layer $\mathcal{L}_{h-1}$ in Phase 5.

We shall first consider the security of the subgroup keys $K_{v,j}$ for $1 \leq v \leq h-1$ and $0 \leq j \leq c^v - 1$ and the final group key K in the up flow information (Phase 1, 2a, 3 and 4).

Theorem 25 *Given the up flow information (Phase 1, 2a, 3 and 4), an adversary $\mathcal{A}$ is unable to obtain the subgroup keys $K_{v,j}$ for $1 \leq v \leq h-1$ and $0 \leq j \leq c^v - 1$ and the final group key K in our*

authenticated proposed protocol provided the BD, NLKW and DSA signature scheme are secure.

Proof: In this proof, we consider two cases, that is, (a) the adversary $\mathcal{A}$ tries to impersonate user $S_{h-1,j}$ ($0 \leq j \leq c^h - 1$) or subgroup controller $S_{v,j}$ ($0 \leq v \leq h - 2$ and $0 \leq j \leq c^{v+1} - 1$) to establish the subgroup keys $K_{v,j}$ ($1 \leq v \leq h - 1$ and $0 \leq j \leq c^v - 1$) and the final group key K , and (b) the adversary $\mathcal{A}$ tries to obtain subgroup keys $K_{v,j}$ ($1 \leq v \leq h - 1$ and $0 \leq j \leq c^v - 1$) and the final group key K from the up flow information sent in Phase 1, 2a, 3 and 4. For case (a), suppose the adversary $\mathcal{A}$ tries to impersonate a user $S_{h-1,j}$ ($0 \leq j \leq c^h - 1$) in Phase 1, then $\mathcal{A}$ computes $z_{h-1,j} = g^{r_{h-1,j}} \bmod p$. Next, $\mathcal{A}$ needs to compute the signature $\sigma_{h-1,j}$ of $S_{h-1,j} || z_{h-1,j}$. The validity of signature $\sigma_{h-1,j}$ is checked by its subgroup controller $S_{h-2,i}$ where $i = \lfloor \frac{j}{c} \rfloor$. If signature $\sigma_{h-1,j}$ is not correct, the $S_{h-2,i}$ rejects $S_{h-1,j}$. As $\mathcal{A}$ does not know the private key of $S_{h-1,j}$ and the DSA signature scheme is assumed to be secure, therefore $\mathcal{A}$ is not able to produce a valid signature $\sigma_{h-1,j}$. Hence $\mathcal{A}$ is not able to impersonate user $S_{h-1,j}$ for $0 \leq j \leq c^h - 1$. Next, suppose the adversary $\mathcal{A}$ tries to impersonate a subgroup controller $S_{v,j}$ ($0 \leq v \leq h - 2$ and $0 \leq j \leq c^{v+1} - 1$) in Phase 2a, then $\mathcal{A}$ computes $\bar{z}_{v,j} = g^{r_{v,j}} \bmod p$, DH keys $T_{v+1,jc+k} = z_{v+1,jc+k}^{r_{v,j}} \bmod p$ for $0 \leq k \leq c - 1$, subgroup key $K_{v+1,j} = \bar{z}_{v,j}^{r_{v,j}} \cdot \prod_{k=0}^{c-1} T_{v+1,jc+k} \bmod p$, ciphertexts $c_{v+1,jc+k} = \mathcal{E}_{T_{v+1,jc+k}}(K_{v+1,j} || S_{v,j})$ for $0 \leq k \leq c - 1$ and $z_{v,j} = g^{H(K_{v+1,j})} \bmod p$. Next, $\mathcal{A}$ has to compute the signatures $\sigma_{v,j}$ and $\bar{\sigma}_{v,j}$ of $S_{v,j} || z_{v,j}$ and $S_{v,j} || \bar{z}_{v,j} || c_{v+1,jc+k}$ for $0 \leq k \leq c - 1$ respectively. The validity of $\sigma_{v,j}$ and $\bar{\sigma}_{v,j}$ ($1 \leq v \leq h - 2$ and $0 \leq j \leq c^{v+1} - 1$) is checked by $S_{v-1,i}$ where $i = \lfloor \frac{j}{c} \rfloor$ and $S_{v+1,jc+k}$ for $0 \leq k \leq c - 1$ respectively. If the signatures $\sigma_{v,j}$ and $\bar{\sigma}_{v,j}$ are not correct, then $S_{v-1,i}$ where $i = \lfloor \frac{j}{c} \rfloor$ and $S_{v+1,jc+k}$ for $0 \leq k \leq c - 1$ rejects $S_{v,j}$. The validity of $\sigma_{0,j}$ and $\bar{\sigma}_{0,j}$ ($0 \leq j \leq c - 1$) is checked by $S_{0,i}$ for $i = j - 1, j + 1$ and $S_{1,jc+k}$ for $0 \leq k \leq c - 1$ respectively. If the signatures $\sigma_{0,j}$ and $\bar{\sigma}_{0,j}$ are not correct, then $S_{0,i}$ for $i = j - 1, j + 1$ and $S_{1,jc+k}$ for $0 \leq k \leq c - 1$ rejects $S_{0,j}$. As $\mathcal{A}$ does not know the private key of $S_{v,j}$ for $0 \leq v \leq h - 2$ and $0 \leq j \leq c^{v+1} - 1$ and the DSA signature scheme is assumed to be secure, therefore $\mathcal{A}$ is not able to produce valid signatures $\sigma_{v,j}$ and $\bar{\sigma}_{v,j}$ for $0 \leq v \leq h - 2$ and $0 \leq j \leq c^{v+1} - 1$. Hence $\mathcal{A}$ is not able to impersonate subgroup controller $S_{v,j}$ for $0 \leq v \leq h - 2$ and $0 \leq j \leq c^{v+1} - 1$ in Phase 2a. Next, suppose the adversary $\mathcal{A}$ tries to impersonate $S_{0,j}$ for $0 \leq j \leq c - 1$ in Phase 3. Then $\mathcal{A}$ computes $X_{0,j} = (z_{0,j+1}/z_{0,j-1})^{H(K_{1,j})}$. Next, $\mathcal{A}$ has to compute signature $\sigma'_{0,j}$ of $S_{0,j} || X_{0,j}$. The validity of $\sigma'_{0,j}$ is checked by $S_{0,i}$ for $0 \leq i \neq j \leq c - 1$. If the signature $\sigma'_{0,j}$ is not correct, then $S_{0,i}$ for $0 \leq i \neq j \leq c - 1$ rejects $S_{0,j}$. As $\mathcal{A}$ does not know

the private key of $S_{0,j}$ and the DSA signature scheme is assumed to be secure, hence $\mathcal{A}$ is not able to produce valid signature $\sigma'_{0,j}$. Hence $\mathcal{A}$ is not able to impersonate $S_{0,j}$ for $0 \leq j \leq c-1$ in Phase 3. Therefore, the adversary $\mathcal{A}$ is not able to impersonate user $S_{h-1,j}$ for $0 \leq j \leq c^h-1$ or subgroup controller $S_{v,j}$ for $0 \leq v \leq h-2$ and $0 \leq j \leq c^{v+1}-1$ to establish the subgroup keys $K_{v,j}$ for $1 \leq v \leq h-1$ and $0 \leq j \leq c^v-1$ and the final group key K . Now consider case (b) as follows: Suppose the adversary $\mathcal{A}$ tries to obtain the subgroup keys $K_{v,j}$ for $1 \leq v \leq h-1$ and $0 \leq j \leq c^v-1$ from the up flow information sent. Then Phase 1 and Phase 2a (without the signature information sent) is just the NLKW protocol for each subgroup $\mathcal{G}_{v,j}$ ($1 \leq v \leq h-1$ and $0 \leq j \leq c^v-1$) to establish the subgroup key $K_{v,j}$. As the NLKW protocol is provably secure, hence the adversary $\mathcal{A}$ is not able to obtain the subgroup key $K_{v,j}$ for $1 \leq v \leq h-1$ and $0 \leq j \leq c^v-1$ using the up flow information. Next, suppose the adversary $\mathcal{A}$ tries to obtain the final group key K from the up flow information sent. Then Phase 2a(ii), Phase 3 and Phase 4 is just the BD protocol to establish the final group key K . As the BD protocol is provably secure, hence the adversary $\mathcal{A}$ is not able to obtain the final group key K from the up flow information. Therefore, we conclude that it is not possible for the adversary $\mathcal{A}$ to obtain the subgroup keys $K_{v,j}$ for $1 \leq v \leq h-1$ and $0 \leq j \leq c^v-1$ and the final group key K from the up flow information sent (Phase 1, 2a, 3 and 4). ■

Next, we shall examine the security of the subgroup keys $K_{v,j}$ and final group key K from the down flow information (Phase 2b and 5).

Theorem 26 *Given the down flow information (Phase 2b and 5), an adversary $\mathcal{A}$ is not able to obtain the subgroup keys $K_{v,j}$ for $1 \leq v \leq h-1$ and $0 \leq j \leq c^v-1$ and final group key K provided the symmetric key encryption scheme SKE is secure and the CDH problem is difficult.*

Proof: As it is already shown in Theorem 25 that the adversary $\mathcal{A}$ is not able to impersonate any user $S_{h-1,j}$ for $0 \leq j \leq c^h-1$ and subgroup controller $S_{v,j}$ for $0 \leq v \leq h-2$ and $0 \leq j \leq c^{v+1}-1$, hence the adversary only has the down flow information in Phase 2b and 5. First, we consider if the adversary $\mathcal{A}$ is able to obtain the subgroup keys $K_{v,j}$ from the down flow information, that is, $\bar{z}_{v-1,j}$, $z_{v,k}$ and $c_{v,k}$ for $1 \leq v \leq h-1$, $0 \leq k \leq c^{v+1}-1$ and $j = \lfloor \frac{k}{c} \rfloor$, in Phase 2b. If the adversary $\mathcal{A}$ is able to obtain $T_{v,k} = g^{r_{v-1,j}H(K_{v+1,k})}$ from $\bar{z}_{v-1,j} = g^{r_{v-1,j}}$ and $z_{v,k} = g^{H(K_{v+1,k})}$ for $1 \leq v \leq h-2$ (or $T_{v,k} = g^{r_{v-1,j}r_{v,k}}$ from $\bar{z}_{v-1,j} = g^{r_{v-1,j}}$ and $z_{v,k} = g^{r_{v,k}}$ for $v = h-1$), then the adversary $\mathcal{A}$ can decrypt $c_{v,k}$ ($1 \leq v \leq h-1$ and $0 \leq k \leq c^{v+1}-1$) and obtain subgroup key $K_{v,j}$ where $j = \lfloor \frac{k}{c} \rfloor$.

But it is not possible for $\mathcal{A}$ to obtain $T_{v,k}$ for $1 \leq v \leq h-1$ and $0 \leq k \leq c^{v+1} - 1$ as this is the CDH problem and is assumed to be difficult. Therefore the adversary $\mathcal{A}$ is not able to obtain the subgroup keys $K_{v,j}$ for $1 \leq v \leq h-1$ and $0 \leq j \leq c^v - 1$ from the down flow information. Next, consider if the adversary $\mathcal{A}$ is able to obtain the final group key K from the down flow information, that is, $c'_{v,k}$ for $0 \leq v \leq h-2$ and $0 \leq k \leq c^{v+1} - 1$, in Phase 5. As the final group key K is the plaintext of $c'_{v,k}$, therefore it is not possible for $\mathcal{A}$ to recover K from $c'_{v,k}$ as it is assumed that the symmetric key encryption scheme SKE is secure. Therefore, the adversary $\mathcal{A}$ is not possible to obtain subgroup keys $K_{v,j}$ for $1 \leq v \leq h-1$ and $0 \leq j \leq c^v - 1$ and final group key K from the down flow information (Phase 2b and 5). ■

8.4 Complexity Analysis

In Table 8.1, we compare the computational (number of modular exponentiations, signature generations and verifications and certificate verifications performed) and communication (number of messages and certificates sent and received) costs of five GKA protocols, namely the BD [BD95], NLKW [NLKW05], Nam et al's Hierarchical GKA protocol (NHS) [NKW05], BD-BD (described in Chapter 7) and our proposed protocol. For the NHS [NKW05], the number of subgroups and number of subgroup members are denoted as m and c respectively while the group size, number of users in each subgroup and height of the hierarchical structure used in the BD-BD are denoted as n , c and h respectively. For the BD-BD, the notations $\mathcal{G}_{0,0}$, $\mathcal{G}_{v,j}$ ($1 \leq v \leq h-3$ and $0 \leq j \leq c^v - 1$) and $\mathcal{G}_{h-2,j}$ ($0 \leq j \leq c^{h-2} - 1$) denotes the subgroup members in subgroups $\mathcal{G}_{0,0}$, $\mathcal{G}_{v,j}$ ($1 \leq v \leq h-3$ and $0 \leq j \leq c^v - 1$) and $\mathcal{G}_{h-2,j}$ ($0 \leq j \leq c^{h-2} - 1$) respectively, which comprises of c members each, who are also controllers of subgroups at the next lower layer. As mentioned earlier, the number of users in each subgroup at every layer for our proposed protocol is not fixed and can vary in different subgroups depending on the application requirements. To illustrate our proposed protocol and without loss of generality, we assume that the network is balanced with every subgroup containing the same number of users, c .

In order to achieve authentication in all five GKA protocols, all messages sent and received are signed and verified respectively. The public keys of the signers that are used to verify the received messages are provided in digital certificates. These digital certificates are signed by a Trusted Third

Party (Certification Authority (CA)) and will be verified by each user using the CA's public key before the signer's public key can be used.

Table 8.1: Computational and Communication Cost

		Exp.	Sign Gen	Sign Verif	Cert tx	Cert Rx	Cert Verif	Msgs Sent	Msgs Rx
BD [BD95]		3	2	$n + 1$	1	$n - 1$	$n - 1$	2	$n + 1$
NLKW [NLKW05]	C	$n + 1$	1	$n - 1$	1	$n - 1$	$n - 1$	1	$n - 1$
	User	2	1	1	1	1	1	1	1
NHS [NKW05]	C	$m + 3$	1	m	1	m	m	1	m
	S	$c + 4$	1	$c + 1$	1	$c + 1$	$c + 1$	1	$c + 1$
	User	3	1	2	1	2	2	1	2
BD- BD (Ch.7)	$\mathcal{G}_{0,0}$	$3h$	$2h$	$h(c + 1)$	h	$h(c - 1)$	$h(c - 1)$	$3h - 1$	$h(c + 1)$
	$\mathcal{G}_{v,j}$	$3(h - v)$	$2(h - v)$	B	$h - v$	C	C	A	$B + 1$
	$\mathcal{G}_{h-2,j}$	6	4	$2(c + 1)$	2	$2(c - 1)$	$2(c - 1)$	5	$2c + 3$
	User	3	2	$c + 1$	1	$c - 1$	$c - 1$	2	$c + 2$
Prop. Prot.	$\mathbf{S}_{0,j}$	$c + 5$	3	$2c + 1$	2	$2c - 1$	$2c - 1$	4	$2c + 1$
	$\mathbf{S}_{v,j}$	$c + 4$	2	$c + 1$	2	$c + 1$	$c + 1$	3	$c + 2$
	$\mathbf{S}_{h-1,i}$	2	1	1	1	1	1	1	2

$$A : 2(h - v) + (h - v + 1)$$

$$B : ((h - v)(c + 1))$$

$$C : ((h - v)(c - 1))$$

For NLKW [NLKW05], **C** refers to the controller node.

For NHS [NKW05], **C** and **S** refers to the controller and subgroup controller node respectively.

For BD-BD, for $\mathcal{G}_{v,j}$, $1 \leq v \leq h - 3$ and $0 \leq j \leq c^v - 1$. For $\mathcal{G}_{h-2,j}$, $0 \leq j \leq c^{h-2} - 1$

In our proposed protocol, for $\mathbf{S}_{0,j}$, $0 \leq j \leq c - 1$. For $\mathbf{S}_{v,j}$, $1 \leq v \leq h - 2$ and $0 \leq j \leq c^{v+1} - 1$. For $\mathbf{S}_{h-1,i}$, $0 \leq i \leq c^h - 1$.

From Table 8.1, it can be seen that the users $\mathbf{S}_{h-1,i}$ in our proposed protocol and the users in NLKW [NLKW05] have to perform 2 exponentiations, which is the lowest number of exponentiations. The users in BD [BD95], NHS [NKW05] and BD-BD performs 3 exponentiations each while the subgroup controllers in the BD-BD performs between $3h$ to 6 exponentiations, depending on their positions in the hierarchical structure used in the BD-BD scheme. The subgroup controllers $\mathbf{S}_{0,j}$ in the CG of our proposed protocol performs $c+5$ exponentiations each and subgroup controllers $\mathbf{S}_{v,j}$ for $1 \leq v \leq h - 2$ performs $c + 4$ exponentiations each. The main controller in NHS [NKW05] performs $m + 3$ exponentiations, where m increases as group size n increases while the subgroup controller in NHS [NKW05] performs $c + 4$ exponentiations, where c is constant regardless of group size n . The controller in NLKW protocol [NLKW05] performs n exponentiations, which is the

highest number of exponentiations to be performed.

In terms of signature generations and verifications, users $S_{h-1,i}$ in our proposed protocol and users in NLKW [NLKW05] are the most efficient, performing 1 signature generation and verification each. Although users in NHS [NKW05] also performs 1 signature generation, but 2 signature verifications are required. Both users in BD [BD95] and BD-BD generates 2 signatures each but users in BD [BD95] have to verify $n + 1$ signatures while users in BD-BD are more scalable, verifying $c + 1$ signatures each. The subgroup controllers in BD-BD have to generate between $2h$ to 4 signatures and verify between $h(c + 1)$ to $2(c + 1)$ signatures, depending on their positions in the hierarchical structure used in the BD-BD scheme. The subgroup controllers $S_{0,j}$ in the CG of our proposed protocol has to generate 3 signatures and verify $2c + 1$ signatures each while the subgroup controllers $S_{v,j}$ for $1 \leq v \leq h - 2$ have to generate 2 signatures and verify $c + 1$ signatures each. The main controller and subgroup controllers in NHS [NKW05] are slightly more efficient, requiring 1 signature generation each while verifying m and $c + 1$ signatures respectively. Although the controller in NLKW [NLKW05] requires only 1 signature generation, however $n - 1$ signature verifications are required.

The controller and users of the NLKW protocol [NLKW05], main controller, subgroup controllers and users of NHS [NKW05] and users $S_{h-1,i}$ of our proposed protocol only has to transmit 1 message, however the message sent by controller in NLKW [NLKW05] is a large message containing n 1024-bits information. The message sent by main controller of NHS [NKW05] contains m 1024-bits information while the message sent by subgroup controllers of NHS [NKW05] contains c 1024-bits information. Although the subgroup controllers $S_{0,j}$ and $S_{v,j}$ for $1 \leq v \leq h - 2$ in our proposed protocol transmit 4 and 3 messages each respectively, only two of these messages sent contains c 1024-bits information while the other messages contain one 1024-bits information each. The subgroup controllers in the BD-BD transmit between $3h - 1$ to 5 messages, depending on their position on the hierarchical structure in the BD-BD scheme, where each message has one 1024-bits information while the users in BD-BD transmit 2 messages that contains one 1024-bits information each.

The users in BD [BD95] and controller in NLKW [NLKW05] has to receive $n + 1$ and $n - 1$ messages respectively, which will be large when group size n is large. Although the users in

NLKW [NLKW05] receives only 1 message each, however, this message contains n 1024-bits information. Main controller and subgroup controllers of NHS [NKW05] receives m and $c + 1$ messages respectively. Although each user in NHS [NKW05] receives only 2 messages, but one of these messages contains c 1024-bits information while the second message contains m 1024-bits information. The users $S_{h-1,i}$ in our proposed protocol receives 2 messages each, but only one of these messages contains c 1024-bits information while the other message contains one 1024-bits information only. The subgroup controllers $S_{0,j}$ in our proposed protocol receives $2c + 1$ messages each, where each message contains one 1024-bits information only while the subgroup controllers $S_{v,j}$ for $1 \leq v \leq h - 2$ receives $c + 2$ messages each, where one message contains c 1024-bits information while the other messages contains one 1024-bits information each. Each user in BD-BD receives $c + 2$ messages, each containing one 1024-bits information while the subgroup controllers in BD-BD receive between $h(c + 1)$ and $2c + 3$ messages, each containing one 1024-bits information.

Although users in BD [BD95], controller and users in NLKW [NLKW05], main controller, subgroup controllers and users in NHS [NKW05], users in BD-BD and our proposed protocol transmits only 1 certificate, however, only users in NLKW [NLKW05] and our proposed protocol has to receive and verify 1 certificate each while users in NHS [NKW05] has to receive and verify 2 certificates each and users in BD-BD receives and verifies $c - 1$ certificates each. The subgroup controllers in BD-BD transmit between h and 2 certificates and receive and verify between $hc - 1$ and $2c - 1$ certificates, depending on their position in the hierarchical structure used in the BD-BD scheme. The subgroup controllers $S_{0,j}$ and $S_{v,j}$ for $1 \leq v \leq h - 2$ in our proposed protocol transmits 2 certificates each and receives and verifies $2c - 1$ and $c + 1$ certificates respectively. The main controller and subgroup controllers in NHS [NKW05] receives and verifies m and $c + 1$ certificates each respectively while the users in BD [BD95] and controller in NLKW [NLKW05] has to receive and verify the largest number of certificates ($n - 1$ certificates).

8.5 Energy Analysis

In this section, we perform the computational (modular exponentiation, signature generation and verification and certificate verification) energy costs analysis of the 133MHz SA-1110 “StrongARM” microprocessor described in Section 3.3.1 and the communication (transmission and reception of

messages and certificates) energy costs of the 11Mbps Spectrum24 LA-4121 WLAN card described in Section 3.4.1. In Section 8.5.1, we provide four graphs and a chart which compares the computational, communication, total energy and total energy consumed by our proposed protocol with four other GKA protocols, namely the BD [BD95], NLKW [NLKW05], Nam et al's Hierarchical GKA protocol (NHS) [NKW05], BD-BD (described in Chapter 7) for different number of user nodes $n = 10, 100, 1000$ and 10000 . We then discuss on the results in Section 8.5.2. It should be noted that for the DSA signature scheme [DSA94], a 172-bytes DSA certificate as described in Section 3.4.3 is used. It should also be noted that besides the BD-BD (described in Chapter 7) and our proposed protocol, only the NHS [NKW05] protocol uses a hierarchical group structure while the BD [BD95] and NLKW [NLKW05] protocols do not use any hierarchical group structure.

8.5.1 Energy Consumption Cost Results

Based on the complexity analysis in Table 8.1, the energy costs of the 133MHz SA-1110 "StrongARM" microprocessor taken from Table 3.2 in Section 3.3.1 and the communication (transmission and reception of messages and certificates) energy costs of the 11Mbps Spectrum24 LA-4121 WLAN card taken from Table 3.4 in Section 3.4.1, we obtain the computational (exponentiation, signature generation and verification and certificate verification), transmission of messages and certificates, reception of messages and certificates and total energy costs comparison for all users running the five GKA protocols as shown in Figure 8.1, 8.2, 8.3 and 8.4 respectively. We also show the total energy consumed by each GKA protocol in Figure 8.5. The Y-axis in Figures 8.1, 8.2, 8.3 and 8.4 measures the energy consumed in milliJoules (mJ) while the Y-axis in Figure 8.5 is in Joules (J). The X-axis shows the number of user nodes $n = 10, 100, 1000$ and 10000 .

The computational energy costs in Figure 8.1 were obtained by multiplying the number of exponentiations, signature generations and verifications and certificate verifications in Table 8.1 with the exponentiation, DSA signature generation, DSA signature verification and DSA certificate verification energy costs in Table 3.2. It should be noted that the energy cost for DSA certificate verification is the same as DSA signature verification. The transmission costs in Figure 8.2 were obtained by multiplying the transmission cost per bit ($0.658\mu J$) and the transmission cost of a DSA certificate ($0.905mJ$) with the total number of bits in each message and total number of certificates

transmitted respectively. Similarly, the reception costs in Figure 8.3 were obtained by multiplying the reception cost per bit ($0.305\mu J$) and reception cost of a DSA certificate ($0.42mJ$) in Table 3.4 with the total number of bits in each message and total number of certificates shown in Table 8.1.

In all the figures, when $n = 10$, we will assume that the NHS protocol [NKW05] and our proposed protocol are equivalent to the NLKW protocol [NLKW05] due to the small number of nodes to form a hierarchical structure. Therefore, the subgroup controllers in NHS [NKW05] are considered as users. Similarly, there is only one subgroup controller in our proposed protocol and the main controller in NHS [NKW05] is the controller in NLKW [NLKW05] when $n = 10$. We also assume that the BD-BD is equivalent to the BD [BD95] when $n = 10$, i.e. there are no subgroup controllers in BD-BD when $n = 10$. For the BD-BD, the notations $\mathcal{G}_{0,0}$ and $\mathcal{G}_{v,j}$ ($1 \leq v \leq 2$ and $0 \leq j \leq c^v - 1$) denotes the subgroup members in subgroups $\mathcal{G}_{0,0}$ and $\mathcal{G}_{v,j}$ ($1 \leq v \leq 2$ and $0 \leq j \leq c^v - 1$) respectively.

For our proposed protocol, the number of users in each subgroup at every layer is not fixed and can vary in different subgroups depending on the application requirements. For illustration purposes in our energy analysis, we assume that the network is balanced with every subgroup containing the same number of users, $c = 10$, for number of user nodes $n = 100, 1000$ and 10000 .

Computational Costs

Figure 8.1 shows the total computational (modular exponentiation, signature generation and verification and certificate verification) energy costs of nodes running the five protocols. It can be seen that the users $S_{h-1,i}$ in our proposed protocol and the users in NLKW protocol [NLKW05] consume the least energy ($18.2mJ$), regardless of the number of user nodes. This is followed by the users in NHS [NKW05] while the users in BD-BD have a higher computational energy costs due to the larger ($\mathcal{O}(c)$) number of signature and certificate verifications required. Although the subgroup controllers $S_{0,j}$ in the CG of our proposed protocol have higher energy costs than the subgroup controllers in NHS [NKW05], however the main controller in NHS [NKW05] have the third highest computational energy costs when $n > 100$. The controller of the NLKW protocol [NLKW05] consumes the most energy while the users in BD [BD95] consumes the second highest energy.

Figure 8.1: Computational Energy Consumption Costs.

Transmission (Communication) Costs

Figure 8.2: Transmission Energy Consumption Costs.

Figure 8.2 shows the transmission (both messages and certificates) energy costs of each node running the five protocols. From the figure, it can be seen that the users $S_{h-1,i}$ in our proposed protocol, users in NLKW [NLKW05] and users in NHS [NKW05] consume the least energy while

the users in the BD [BD95] and BD-BD consume the second lowest amount of energy. Although subgroup controllers in NHS [NKW05] consume lower amount of energy than the subgroup controllers $S_{0,j}$ in our proposed protocol, however main controller of NHS [NKW05] consumes the second highest amount of energy when $n > 100$ while controller in NLKW [NLKW05] consumes the highest energy for transmission of messages and certificates.

Reception (Communication) Costs

Figure 8.3: Reception Energy Consumption Costs.

Figure 8.3 shows the reception (both messages and certificates) energy costs of each node running the five protocols. it can be seen that the users $S_{h-1,i}$ in our proposed protocol consume the least energy. This is followed by the users in BD-BD, the subgroup controllers $S_{v,j}$ for $1 \leq v \leq h-2$ and $S_{0,j}$ in our proposed protocol and the subgroup controllers in BD-BD. The subgroup controllers $S_{0,j}$ in our proposed protocol are more efficient than the subgroup controllers and users in NHS [NKW05] when $n > 100$. The main controller of NHS [NKW05] has the fourth highest energy costs followed by the controller and users in NLKW [NLKW05].

Figure 8.4: Total Energy Consumption Costs.

Total Energy Costs

Figure 8.4 shows the total (both computational and communication) energy costs of each node running the five protocols. From the figure, it can be seen that the users $S_{h-1,i}$ in our proposed protocol are the most efficient, consuming the lowest energy. This is followed by users in NHS [NLKW05]. When $n > 100$, the users in NLKW [NLKW05] becomes inefficient due to the large amount of message reception. The users in the BD-BD are also less efficient than the users $S_{h-1,i}$ in our proposed protocol and users in NHS [NLKW05]. Although the subgroup controllers $S_{0,j}$ of our proposed protocol are less efficient than the subgroup controllers of NHS [NLKW05] but the main controller in NHS [NLKW05] has the third highest energy costs when $n > 100$. Controller in NLKW [NLKW05] has the highest energy costs for GKA followed by the users in the BD [BD95].

Total Energy Consumed by each GKA protocol

Figure 8.5 shows the total energy costs by each of the five GKA protocols, namely BD [BD95], NLKW [NLKW05], NHS [NLKW05], BD-BD and our proposed protocol. The total energy costs in Figure 8.5 were obtained by adding up the total energy consumed by every node in each respective GKA protocol.

From Figure 8.5, we can see that our proposed protocol is the most energy-efficient for different

Figure 8.5: Total Energy Consumption Costs for each GKA.

number of user nodes $n = 10, 100, 1000$ and 10000 . The BD [BD95] consumes the most energy for $n = 10, 100, 1000$ and 10000 due to the large amounts of signature and certificate verifications required as well as the large number of messages and certificates to be received. When the number of user nodes n is less than 100, the BD-BD scheme consumes the second highest energy because the users in BD-BD scheme have to verify larger amount of signatures and certificates which results in much higher computational energy costs as compared to the users in NLKW [NLKW05]. When the number of user nodes n is more than 100, the NLKW [NLKW05] consumes the second highest amount of energy because the users in NLKW [NLKW05] has to receive a larger message as the number of user nodes increases which results in much higher reception energy costs as compared to the users in the GKA protocol for the (C-H) group model.

8.5.2 Discussion

In HWNs, the GKA protocol used in these networks should result in low energy costs for user nodes. From the energy analysis, it can be seen that the users $S_{h-1,i}$ in our proposed protocol have the lowest energy costs as compare to the BD [BD95], NLKW [NLKW05], NHS [NKW05] and BD-BD (the GKA protocol for the (C-H) group model described in Chapter 7). As mentioned earlier, besides the BD-BD and our proposed protocol, only the NHS [NKW05] protocol uses a hierarchical group structure while the BD [BD95] and NLKW [NLKW05] protocols do not use any hierarchical group structure. Although the subgroup controllers $S_{0,j}$ in our proposed protocol

consume slightly higher amount of energy than the subgroup controllers in NHS [NKW05], however it should be noted that the main controller in NHS [NKW05] consumes the third highest amount of energy as shown in Figure 8.4 when $n > 100$. Furthermore, the users in NHS [NKW05] consumes more energy than the users $S_{h-1,i}$ in our proposed protocol regardless of the number of user nodes in the network. Figure 8.5 also shows that the total energy consumed by our proposed protocol is the lowest as compared to the other four GKA protocols regardless of the number of user nodes in the network. Hence, our proposed protocol is more energy-efficient in HWNs where there are low-power user nodes with energy constraints and high-power nodes without energy constraints.

8.6 Summary

GKA protocols designed for HWNs have to be energy-efficient and scalable. Authentication in GKA protocol is also important to prevent impersonation and man-in-the-middle attacks. In this chapter, we proposed an energy-efficient and scalable authenticated GKA protocol for HWNs, which uses the ContRi-Central group model. A complexity analysis and energy analysis indicate that our proposed protocol is more efficient and suitable for use in HWNs than four other GKA protocols.

Chapter 9

Denial-of-Service Attack Resilience Dynamic Group Key Agreement for Heterogeneous Networks

9.1 Introduction

GKA protocols in heterogenous networks are prone to Denial-of-Service (DoS) attacks where the malicious adversary may bring down GKA services by launching many invalid `Start_GKA` or `Join` requests. This will cause honest nodes to repeatedly restart the GKA protocol, which consist of CPU-intensive cryptographic operations such as modular exponentiations, without establishing any group keys. Hence, to ensure the availability of GKA services for secure group communications, it is important for GKA protocols to provide protection against DoS attacks. However, most current GKA protocols in literature do not discuss on protection against DoS attacks.

Besides DoS attacks, malicious group members also known as insider adversaries [DPSW06] can also launch insider attacks in which they send “bad” or “wrong” protocol messages to halt the GKA protocol. This will cause all group members to repeatedly re-start the GKA protocol, thereby depleting system resources without establish any group key. Kim et al [KLL04] briefly described in their paper that it is not easy to detect insider adversaries but did not go into details on how to recover from such an insider attack. In fact, most current GKA protocols, with the exception of [DPSW06], focus mainly on protection against malicious adversaries (also known as malicious outsider adversaries) and do not consider malicious group members.

In this chapter, we propose three GKA protocols, namely the Initial GKA, Join and Leave

Protocol, that are based on Kim's GKA protocol [KLL04] and NLKW protocol [NLKW05] to obtain the subgroup keys K_i and final group key K for the Central-ContRi Group Model (described in Section 2.12.3 of Chapter 2) and also change the key of the subgroup that users are joining or leaving and the final group key when dynamic group membership events such as user join or user leave occur. The proposed protocols provide both protection against insider and outsider DoS attacks and key confirmation, which are not provided by most GKA protocols in current literature. We also propose a detection protocol which is able to detect malicious group members that try to disrupt GKA services by launching insider attacks. After detecting the malicious group members, the proposed GKA protocols can continue to generate the group keys without halting and restarting the GKA, thus preventing the insider attack launched by the malicious group members. Both the complexity and energy analysis show that the proposed Initial GKA, Join and Leave Protocol are more energy efficient and scalable despite providing protection against both malicious insider and outsider DoS attacks and key confirmation properties. Furthermore, our communication energy analysis separates point-to-point (P2P) and broadcast communications (transmission and reception) to provide a more detailed study of communication energy costs as compared to current energy analysis which do not separate energy per bit calculations for P2P and broadcast transmission and reception energy costs. We also state the assumptions for the Certification Authority and Heterogeneous Network nodes below.

9.1.1 Certification Authority

We assume that a Public Key Infrastructure (PKI) is in place in the heterogeneous network. All network nodes taking part in the GKA should first select its private and public key pair and register its public key with the Certification Authority (CA) to obtain a RSA public key certificate through a secure channel during the Initialization Phase.

9.1.2 Heterogeneous Network nodes

We assume that the asymmetric heterogeneous network consists of high power nodes (e.g. access points and mesh routers) without mobility, have no energy constraints and do not leave the network unless they are faulty. The user nodes in the heterogeneous network are assumed to be low power

energy constrained devices such as PDAs and laptop computers that can join or leave the network at any time. Besides malicious group outsiders, we also allow the possibility of malicious group members in the heterogeneous network that try to disrupt and halt the GKA services. However, we assume that the malicious group members are independent and do not collaborate with each other. We also assume that there is hardware protection for the high power nodes in the heterogeneous network. Therefore, attacks such as tampering, capture or replication of these high power nodes will not be possible as they are physically protected.

9.2 Proposed Group Key Agreement Protocols for the Central-ContRi Group Model

In this section, we present the three proposed authenticated GKA protocols, namely the Initial Group Key Agreement (GKA) (Group Formation) Protocol, Join Protocol and Leave Protocol, for the Central-ContRi Group Model described in Section 2.12.3 of Chapter 2. Without loss of generality, we assume for our GKA protocols that there are only two layers in the Central-ContRi group model. We refer readers to Section 2.12.3 in Chapter 2 for more details of the Central-ContRi Group Model.

The Initial GKA Protocol will be executed first to establish the subgroup keys K_i and the final group key K for a heterogenous network of n nodes. The Join Protocol will be executed whenever new users join the network in order to change the subgroup key of the subgroup that the new users join and also change the final group key to protect previous secure group communications (backward secrecy) from the new users. The Leave Protocol will be executed whenever users leave the network in order to change the subgroup key of the subgroup that the users left and also change the final group key to protect future secure group communications (forward secrecy) from the leaving users.

During the initialization phase, all nodes will compute their respective RSA private key d_x and public key $(\bar{N}_x, e)$ where $x = \mathbb{C}, i, (i, j)$ using the key generation phase of the RSA scheme shown in Section 2.3.4, and register and obtain the RSA certificate $\text{cert}[x]$ where $x = \mathbb{C}, \mathbb{S}_i, \mathbb{U}_{i,j}$ from the Certification Authority. It should be noted that in the public keys $(\bar{N}_x, e)$ where $x = \mathbb{C}, i, (i, j)$,

the element $e = 2^{16} + 1 = 65537$ is common while $\bar{N}_x$ is different for all nodes. Next, two random large primes, q and p , are selected such that q divides $p - 1$. (p is 1024 bits and q is 160 bits.) An element $g \in Z_p^*$ of order q is selected. Let H be a cryptographic hash function. The index v , which starts from 0 in the Initial GKA protocol, will increase whenever a group membership event such as user join or user leave occurs. We also require a symmetric key encryption scheme **SKE** for the proposed protocols. In the following three GKA protocols, denote subgroup controllers as $S_i = U_{i,1}$, $c = \mathcal{E}_k(m)$ as ciphertext c of message m encrypted with key k , $\sigma = \mathcal{S}(m)$ as signature of message m signed where $\mathcal{S}$ refers to the RSA signature scheme and $H(m)$ as a hashed message digest of message m . It should be noted that in the three proposed GKA protocols, upon receiving both the signature and certificate, the node $(C, S_i \text{ or } U_{i,j})$ performing the signature verifications would first uses the public key in the certificate to verify the signature. If signature is not valid, then there is no need to verify the certificate. On the other hand, if the certificate is verified first (as in the conventional way), then the DoS attacker can send a valid certificate, but invalid signature. In this way, the node $(C, S_i \text{ or } U_{i,j})$ performing the signature verifications would have to spent extra time to verify the certificate, which is correct and then verify the signature and discover that the signature is not valid. We now present the three proposed protocols in detail.

9.2.1 Initial Group Key Agreement

The initial GKA protocol will establish subgroup keys K_i for each subgroup $\mathcal{G}_i$ for $1 \leq i \leq n_c$ and a final group key K for the entire heterogenous network. The final group key K can be used for secure group communications in the heterogenous network since all users know K while the subgroup key K_i can be used for secure subgroup communications within subgroup $\mathcal{G}_i$ since only subgroup members of subgroup $\mathcal{G}_i$ will have the same subgroup key K_i . This is useful in some situations where a subgroup wishes to communicate securely only within its subgroup and not allow other subgroups to listen into its secure communications.

The Initial GKA consists of 3 phases. In Phase 1, the subgroup controllers $S_i = U_{i,1}$ of subgroup $\mathcal{G}_i$ for $1 \leq i \leq n_c$ will interact with its user nodes $U_{i,j}$ for $2 \leq j \leq n_i$ and compute the respective subgroup key K_i . Next in Phase 2, the subgroup controllers S_i will use their respective subgroup keys K_i to interact with the main controller C to compute the final group key K . Finally in Phase 3,

the final group key K is disseminated from the main controller $\mathcal{C}$ to all user nodes $\mathcal{U}_{i,j}$ via subgroup controllers $\mathcal{S}_i$. Key confirmation messages are also sent together to verify and confirm the subgroup keys K_i and final group key K . Note that counter $v = 0$ for the Initial GKA protocol. The counter v is used to keep track of the number of protocol executions and also to be used as an input to the hash functions to prevent replay attacks. The Initial GKA protocol is as follows:

Phase 1: Subgroup Key Generation

Round 1 : Each subgroup controller $\mathcal{S}_i = \mathcal{U}_{i,1}$ for $1 \leq i \leq n_c$ randomly selects $\kappa_{i,1} \in \{0,1\}^\lambda$, computes $\tau_i = H(\kappa_{i,1}||0)$ and broadcasts $\{\mathcal{S}_i||\tau_i\}$ to subgroup members of $\mathcal{G}_i$, where λ is an integer.

Round 2 : Each subgroup member $\mathcal{U}_{i,j}$ for $1 \leq i \leq n_c$ and $2 \leq j \leq n_i$ randomly selects $r_{i,j} \in Z_q^*$ and computes $z_{i,j} = g^{r_{i,j}} \bmod p$. Next, $\mathcal{U}_{i,j}$ broadcasts $\{\mathcal{U}_{i,j}||z_{i,j}||\sigma_{i,j}||\text{cert}[\mathcal{U}_{i,j}]\}$, where $\sigma_{i,j} = \mathcal{S}(\mathcal{U}_{i,j}||z_{i,j}||\tau_i||0)$ and stores τ_i .

Round 3 : Each $\mathcal{S}_i$ for $1 \leq i \leq n_c$ first checks the received signatures $\sigma_{i,j}$ for $2 \leq j \leq n_i$. If the signature verifications are correct, $\mathcal{S}_i$ checks the received certificates $\text{cert}[\mathcal{U}_{i,j}]$ for $2 \leq j \leq n_i$. If $\mathcal{S}_i$ detects invalid certificates and/or signatures sent by certain malicious nodes, then these malicious nodes will be placed into a list BL_Z . $\mathcal{S}_i$ then selects random $r_{i,1} \in Z_q^*$, computes $z_{i,1} = g^{r_{i,1}} \bmod p$ and broadcasts $\{\mathcal{S}_i||z_{i,1}||\text{BL}_Z||\sigma_{i,1}||\text{cert}[\mathcal{S}_i]\}$, where $\sigma_{i,1} = \mathcal{S}(\mathcal{S}_i||\tau_i||z_{i,1}||\text{BL}_Z||0)$.

Round 4 : Each $\mathcal{U}_{i,j}$ for $1 \leq i \leq n_c$ and $2 \leq j \leq n_i$ first checks the received signature $\sigma_{i,1}$ and certificate $\text{cert}[\mathcal{S}_i]$. If the verifications are not correct, then $\mathcal{U}_{i,j}$ rejects $\mathcal{S}_i$. Otherwise, $\mathcal{U}_{i,j}$ checks the list BL_Z to identify the malicious nodes to be blocked from the GKA. (For example, if $\mathcal{U}_{i,j}$ discovers that its neighbor $\mathcal{U}_{i,j-1}$ is in the list BL_Z , then $\mathcal{U}_{i,j}$ will excludes neighbor $\mathcal{U}_{i,j-1}$ from GKA and set $\mathcal{U}_{i,j-2}$ as its neighbor i.e. $\mathcal{U}_{i,j-2} = \mathcal{U}_{i,j-1}$, provided $\mathcal{U}_{i,j-2}$ is not in BL_Z). Next, $\mathcal{U}_{i,j}$ computes $L_{i,j} = H(z_{i,j-1}^{r_{i,j}}||\text{SID}_i)$, $R_{i,j} = H(z_{i,j+1}^{r_{i,j}}||\text{SID}_i)$ and generates $X_{i,j} = L_{i,j} \oplus R_{i,j}$ where $\text{SID}_i = H(\tau_i||\mathcal{U}_{i,1}||\mathcal{U}_{i,2}||\dots||\mathcal{U}_{i,n_i})$. Finally, $\mathcal{U}_{i,j}$ randomly selects $\kappa_{i,j} \in \{0,1\}^\lambda$ and broadcasts $\{\mathcal{U}_{i,j}||\kappa_{i,j}||X_{i,j}||\bar{\sigma}_{i,j}\}$, where $\bar{\sigma}_{i,j} = \mathcal{S}(\mathcal{U}_{i,j}||\kappa_{i,j}||X_{i,j}||\tau_i||0)$.

Round 5 : Each $\mathcal{S}_i$ for $1 \leq i \leq n_c$ first checks the received signatures $\bar{\sigma}_{i,j}$ for $2 \leq j \leq n_i$. If there are invalid signatures, $\mathcal{S}_i$ will place the identities of invalid users $\tilde{\mathcal{U}}_{i,j}$ that generated the invalid signature to a list BL_X and broadcast $\{\mathcal{S}_i||\text{BL}_X||\bar{\sigma}_{i,1}\}$, where $\bar{\sigma}_{i,1} = \mathcal{S}(\mathcal{S}_i||\tau_i||\text{BL}_X||0)$. Otherwise, $\mathcal{S}_i$ computes $L_{i,1} = H(z_{i,n_i}^{r_{i,1}}||\text{SID}_i)$, $R_{i,1} = H(z_{i,2}^{r_{i,1}}||\text{SID}_i)$, $X_{i,1} = L_{i,1} \oplus R_{i,1}$ and $\widehat{X}_i = \kappa_{i,1} \oplus R_{i,1}$. Next, $\mathcal{S}_i$ checks the XOR equation: $X_{i,1} \oplus X_{i,2} \oplus \dots \oplus X_{i,n_i} = 0$. If the XOR equation is not correct, $\mathcal{S}_i$ will place

the invalid XOR equation to list BL_X and broadcast $\{S_i || \text{BL}_X || \bar{\sigma}_{i,1}\}$, where $\bar{\sigma}_{i,1} = \mathcal{S}(S_i || \tau_i || \text{BL}_X || 0)$. Otherwise, if the XOR equation is correct, then S_i broadcast $\{S_i || \widehat{X}_i || X_{i,1} || \bar{\sigma}_{i,1}\}$, where $\bar{\sigma}_{i,1} = \mathcal{S}(S_i || \tau_i || \widehat{X}_i || X_{i,1} || 0)$.

Round 6 :

Case 1 - $U_{i,j}$ receives $\{S_i || \widehat{X}_i || X_{i,1} || \bar{\sigma}_{i,1}\}$: If $U_{i,j}$ for $1 \leq i \leq n_c$ and $2 \leq j \leq n_i$ receives $\{S_i || \widehat{X}_i || X_{i,1} || \bar{\sigma}_{i,1}\}$ from S_i , it means that S_i has verified that all signatures $\bar{\sigma}_{i,k}$ for $2 \leq k \leq n_i$ are valid and the XOR equation is correct in Round 5. In this case, $U_{i,j}$ verifies received signature $\bar{\sigma}_{i,1}$. If the verification is not correct, $U_{i,j}$ rejects S_i . Otherwise, $U_{i,j}$ computes

$$\begin{aligned} \bar{R}_{i,j+1} &= X_{i,j+1} \oplus R_{i,j} \\ \bar{R}_{i,j+2} &= X_{i,j+2} \oplus \bar{R}_{i,j+1} \\ &\vdots \\ \bar{R}_{i,j-1} &= X_{i,j-1} \oplus \bar{R}_{i,j-2} \end{aligned}$$

and compares $\bar{R}_{i,j-1}$ with $L_{i,j}$. If $\bar{R}_{i,j-1} = L_{i,j}$, then all $U_{i,j}$ have $R_{i,1}$ and uses $R_{i,1}$ to obtain $\bar{\kappa}_{i,1}$ from $\widehat{X}_i$. Next, $U_{i,j}$ checks if $H(\bar{\kappa}_{i,1} || 0) = \tau_i$ holds. If yes, all $U_{i,j}$ and S_i computes the subgroup key K_i as follows:

$$K_i = H(\kappa_{i,1} || \kappa_{i,2} || \cdots || \kappa_{i,n_i} || 0) \quad (9.1)$$

Case 2 - $U_{i,j}$ receives $\{S_i || \text{BL}_X || \bar{\sigma}_{i,1}\}$: If $U_{i,j}$ for $1 \leq i \leq n_c$ and $2 \leq j \leq n_i$ receives $\{S_i || \text{BL}_X || \bar{\sigma}_{i,1}\}$ from S_i , it means that S_i has detected either invalid signatures or invalid XOR equation in Round 5. In this case, $U_{i,j}$ first verifies the received signature $\bar{\sigma}_{i,1}$. If the verification is not correct, $U_{i,j}$ rejects S_i . Otherwise, $U_{i,j}$ examines the list BL_X to determine if S_i detected invalid signatures or invalid XOR equation in Round 5.

- [Invalid Signatures] If S_i detected invalid signatures in Round 5, then list BL_X contains the identities of invalid users $\tilde{U}_{i,j}$ for $2 \leq j \leq n_i$ that sent the invalid signature $\bar{\sigma}_{i,j}$. Each $U_{i,j}$ then discards the invalid users $\tilde{U}_{i,j}$ from the GKA.
- [Invalid XOR Equation] If S_i detected an invalid XOR equation, then list BL_X contains the

invalid XOR equation instead of invalid users' identities. The detection protocol described in Section 9.3 will be used to detect and block malicious group members $\tilde{U}_{i,j}$ that sent wrong information to disrupt the GKA.

For brevity, we shall use the term $\tilde{U}_{i,j}$ to denote invalid users and malicious group members. After detecting all invalid users and malicious group members $\tilde{U}_{i,j}$, the honest left neighbor $U_{i,j-1}$, which has not been blocked, of $\tilde{U}_{i,j}$ will re-compute $L_{i,j-1} = H(z_{i,j-2}^{r_{i,j-1}} || \widetilde{SID}_i)$, $R_{i,j-1} = H(z_{i,j+1}^{r_{i,j-1}} || \widetilde{SID}_i)$, $X_{i,j-1} = L_{i,j-1} \oplus R_{i,j-1}$ and transmit $X_{i,j-1}$ with signature, where $\widetilde{SID}_i$ is equal to SID_i excluding the identities of the discarded users in BL_X . The honest right neighbor $U_{i,j+1}$, which has not been blocked, of $\tilde{U}_{i,j}$ will re-compute $L_{i,j+1} = H(z_{i,j-1}^{r_{i,j+1}} || \widetilde{SID}_i)$, $R_{i,j+1} = H(z_{i,j+2}^{r_{i,j+1}} || \widetilde{SID}_i)$, $X_{i,j+1} = L_{i,j+1} \oplus R_{i,j+1}$ and transmit $X_{i,j+1}$ with signature. It should be noted that if left neighbor $U_{i,j-1}$ of $\tilde{U}_{i,j}$ is also an invalid user or malicious group member that has been blocked, then we set $U_{i,j-2} = U_{i,j-1}$. Similarly, if right neighbor $U_{i,j+1}$ of $\tilde{U}_{i,j}$ is also an invalid user or malicious group member that has been blocked, then we set $U_{i,j+2} = U_{i,j+1}$. All other users $U_{i,j}$ will use the new session identity $\widetilde{SID}_i$ to re-compute $L_{i,j}$, $R_{i,j}$, $X_{i,j}$ as in Round 4 and transmit $X_{i,j}$ with signature. Next, Round 5 and Round 6 can continue without invalid signatures and invalid XOR equations detected in Round 5 as all the invalid users and malicious group members $\tilde{U}_{i,j}$ have been detected and blocked.

Phase 2: Final Group Key Generation

Round 1 : Each S_i for $1 \leq i \leq n_c$ computes $y_i = g^{K_i} \bmod p$ and sends $\{S_i || y_i || \sigma_i || \text{cert}[S_i]\}$ to main controller C , where $\sigma_i = \mathcal{S}(S_i || y_i)$.

Round 2 : Main controller C first verifies the received signatures σ_i and certificates $\text{cert}[S_i]$ for $1 \leq i \leq n_c$. If the verifications are not correct, then C rejects S_i . Otherwise, C selects random $r_c \in Z_q^*$ and computes $y_c = g^{r_c} \bmod p$. Next, C computes the Diffie-Hellman (DH) keys $T_i = y_i^{r_c} \bmod p$ that it shares with each S_i . C then computes the final group key $K = (y_c \cdot \prod_{i=1}^{n_c} y_i)^{r_c} \bmod p$ and the key confirmation code $\theta_c = H(K || y_c)$. Next, C encrypts the final group key K using T_i for $1 \leq i \leq n_c$ and obtains $c_i = \mathcal{E}_{T_i}(K)$ for $1 \leq i \leq n_c$. Main controller C then broadcasts $\{C || y_c || c_1 || c_2 || \dots || c_{n_c} || \theta_c || \sigma_c || \text{cert}[C]\}$, where $\sigma_c = \mathcal{S}(C || y_c || \theta_c)$.

Phase 3: Distributing the Final Group Key

Round 1 : Each S_i for $1 \leq i \leq n_c$ first verifies signature σ_C and certificate $\text{cert}[C]$. If the verifications are not correct, then S_i rejects C . Otherwise, S_i computes $T_i = y_C^{K_i} \bmod p$, the DH key it shares with C and uses T_i to decrypt c_i and obtains the final group key K . S_i then confirms and accepts the final group key K if $H(K||y_C) = \theta_C$. Next, S_i encrypts $K||S_i$ using subgroup key K_i and obtains $\bar{c}_i = \mathcal{E}_{K_i}(K||S_i)$. Finally, S_i computes the key confirmation code $\theta_i = H(K_i||K||\tau_i)$ and broadcasts $\{S_i||\bar{c}_i||\theta_i||\bar{\sigma}_i\}$ where $\bar{\sigma}_i = \mathcal{S}(S_i||\tau_i||\theta_i||0)$ to subgroup members in $\mathcal{G}_i$.

Round 2 : Each $U_{i,j}$ for $1 \leq i \leq n_c$ and $2 \leq j \leq n_i$ first verifies the received signature $\bar{\sigma}_i$. If the verification is not correct, then $U_{i,j}$ rejects S_i . Otherwise, $U_{i,j}$ uses its respective subgroup key K_i to decrypt $\bar{c}_i$ and obtains the identity S_i and final group key K . If the identity S_i was not decrypted correctly, then $U_{i,j}$ rejects $\bar{c}_i$. Otherwise, $U_{i,j}$ checks if $\theta_i = H(K_i||K||\tau_i)$. If they are equal, $U_{i,j}$ accepts the final group key K and both K_i and K are confirmed.

9.2.2 Join Protocol

Suppose a set of m new users $\mathcal{J} = \{U_{n+1}, \dots, U_{n+m}\}$ wishes to join subgroup $\mathcal{G}_J = \{U_{J,1}, \dots, U_{J,n_J}\}$. The Join Protocol will be executed to establish a new subgroup key K'_J for subgroup $\mathcal{G}'_J = \mathcal{G}_J \cup \mathcal{J} = \{U_{J,1}, \dots, U_{J,n_J+m}\}$, where $U_{J,j}$ for $1 \leq j \leq n_J$ are the existing users and $U_{J,k}$ for $n_J+1 \leq k \leq n_J+m$ are the new users, and new final group key K' for all users in the network to prevent the new users from knowing previous group communications. The subgroup controller $S_J = U_{J,1}$ will act as the representative of subgroup $\mathcal{G}'_J$ and interact with existing last user U_{J,n_J} and new users $U_{J,k}$ for $n_J+1 \leq k \leq n_J+m$ to establish the new subgroup key K'_J . We let $S_J = U_{J,1}$ be the left neighbor of new user U_{J,n_J+1} and U_{J,n_J} be the right neighbor of the last new user U_{J,n_J+m} for the Join Protocol (i.e. $U_{J,n_J+m+1} = U_{J,n_J}$, $U_{J,n_J+m+2} = U_{J,1}$ and $U_{J,n_J+m+3} = U_{J,n_J+1}$).

The Join Protocol consists of 3 phases. During Phase 1, subgroup controller S_J of subgroup $\mathcal{G}'_J$, U_{J,n_J} and new users $U_{J,k}$ for $n_J+1 \leq k \leq n_J+m$ will interact and compute the new subgroup key K'_J . Next in Phase 2, S_J uses K'_J to interact with main controller C in order for C to compute the new final group key K' . Finally in Phase 3, the new final group key K' is disseminated from C to all user nodes via subgroup controllers S_i for $1 \leq i \leq n_c$ (including S_J). S_J will use new subgroup key K'_J and previous subgroup key K_J to encrypt and send the new final group key K' to new users $U_{J,k}$ for $n_J+1 \leq k \leq n_J+m$ and U_{J,n_J} and the new subgroup key K'_J and new final group key

K' to existing users $U_{J,j}$ for $2 \leq j \leq n_J - 1$ respectively. Key confirmation messages are also sent to verify and confirm the new subgroup key K'_J and new final group key K' . The Join Protocol is described below:

Phase 1: Subgroup Key Generation

Round 1 : Subgroup controller $S_J = U_{J,1}$ randomly selects $\kappa'_{J,1} \in \{0, 1\}^\lambda$, computes $\tau'_J = H(\kappa'_{J,1} || v)$ and broadcasts $\{S_J || \tau'_J || v\}$ to subgroup members $U_{J,j}$ for $2 \leq j \leq n_J$ and new users $U_{J,k}$ for $n_J + 1 \leq k \leq n_J + m$, where λ is an integer and v is a counter that starts from 0 in the Initial GKA Protocol and increment by 1 whenever dynamic GKA protocol such as Join and Leave Protocol is executed.

Round 2 : Each new user $U_{J,k}$ for $n_J + 1 \leq k \leq n_J + m$ and U_{J,n_J} randomly selects $r'_{J,k} \in Z_q^*$ and $r'_{J,n_J} \in Z_q^*$ respectively, and computes $z'_{J,k} = g^{r'_{J,k}} \bmod p$ and $z'_{J,n_J} = g^{r'_{J,n_J}} \bmod p$ respectively. Next, $U_{J,k}$ stores τ'_J and broadcasts $\{U_{J,k} || z'_{J,k} || \sigma'_{J,k} || \text{cert}[U_{J,k}]\}$, where $\sigma'_{J,k} = \mathcal{S}(U_{J,k} || z'_{J,k} || \tau'_J || v)$ and U_{J,n_J} stores τ'_J and broadcasts $\{U_{J,n_J} || z'_{J,n_J} || \sigma'_{J,n_J}\}$, where $\sigma'_{J,n_J} = \mathcal{S}(U_{J,n_J} || z'_{J,n_J} || \tau'_J || v)$.

Round 3 : S_J checks the received signatures $\sigma'_{J,k}$ for $n_J + 1 \leq k \leq n_J + m$ and σ'_{J,n_J} . If the signature verifications are correct, S_J checks the received certificates $\text{cert}[U_{J,k}]$. If S_J detects invalid certificates and/or signatures sent by certain malicious nodes, then these malicious nodes will be placed into a list BL_Z . S_J then selects random $r'_{J,1} \in Z_q^*$ and computes $z'_{J,1} = g^{r'_{J,1}} \bmod p$ and broadcasts $\{S_J || z'_{J,1} || BL_Z || \sigma'_{J,1} || \text{cert}[S_J]\}$, where $\sigma'_{J,1} = \mathcal{S}(S_J || \tau'_J || z'_{J,1} || BL_Z || v)$.

Round 4 :

(i) Each $U_{J,k}$ for $n_J + 1 \leq k \leq n_J + m$ first checks the received signature $\sigma'_{J,1}$ and certificate $\text{cert}[S_J]$. If the verifications are not correct, then $U_{J,k}$ rejects S_J . Otherwise, $U_{J,k}$ checks the list BL_Z to identify the malicious nodes to be blocked from the GKA. (For example, if $U_{J,k}$ discovers that its neighbor $U_{J,k-1}$ is in the list BL_Z , then $U_{J,k}$ will excludes neighbor $U_{J,k-1}$ from GKA and set $U_{J,k-2}$ as its neighbor, i.e. $U_{J,k-2} = U_{J,k-1}$, provided $U_{J,k-2}$ is not in BL_Z). Next, $U_{J,k}$ computes $L'_{J,k} = H((z'_{J,k-1})^{r'_{J,k}} || \text{SID}'_J)$ and $R'_{J,k} = H((z'_{J,k+1})^{r'_{J,k}} || \text{SID}'_J)$, where $z'_{J,n_J+m+1} = z'_{J,n_J}$, $z'_{J,n_J+m+2} = z'_{J,1}$, $z'_{J,n_J+m+3} = z'_{J,n_J+1}$ and $\text{SID}'_J = H(\tau'_J || U_{J,1} || \dots || U_{J,n_J+m} || U_{J,n_J})$. $U_{J,k}$ then generates $X'_{J,k} = L'_{J,k} \oplus R'_{J,k}$. Finally, $U_{J,k}$ randomly selects $\kappa'_{J,k} \in \{0, 1\}^\lambda$ and broadcasts $\{U_{J,k} || \kappa'_{J,k} || X'_{J,k} || \bar{\sigma}'_{J,k}\}$, where $\bar{\sigma}'_{J,k} = \mathcal{S}(U_{J,k} || \kappa'_{J,k} || X'_{J,k} || \tau'_J || v)$.

(ii) U_{J,n_J} first checks signature $\sigma'_{J,1}$. If the verification is not correct, U_{J,n_J} rejects S_J . Otherwise,

U_{J,n_J} checks the list BL_Z to identify the malicious nodes to be blocked from the GKA. Next, U_{J,n_J} computes $L'_{J,n_J} = H((z'_{J,n_J+m})^{r'_{J,n_J}} || SID'_J)$, $R'_{J,n_J} = H((z'_{J,1})^{r'_{J,n_J}} || SID'_J)$ and $X'_{J,n_J} = L'_{J,n_J} \oplus R'_{J,n_J}$. Finally, U_{J,n_J} selects $\kappa'_{J,n_J} \in \{0,1\}^\lambda$ and broadcasts $\{U_{J,n_J} || \kappa'_{J,n_J} || X'_{J,n_J} || \bar{\sigma}'_{J,n_J}\}$, where $\bar{\sigma}'_{J,n_J} = \mathcal{S}(U_{J,n_J} || \kappa'_{J,n_J} || X'_{J,n_J} || \tau'_J || v)$.

Round 5 : S_J first checks the received signatures $\bar{\sigma}'_{J,k}$ for $n_J + 1 \leq k \leq n_J + m$ and $\bar{\sigma}'_{J,n_J}$. If there are invalid signatures, S_J will place the identities of invalid users $\tilde{U}_{J,k}$ that generated the invalid signature to a list BL_X and broadcast $\{S_J || BL_X || \bar{\sigma}'_{J,1}\}$, where $\bar{\sigma}'_{J,1} = \mathcal{S}(S_J || \tau'_J || BL_X || v)$. Otherwise, S_J computes $L'_{J,1} = H((z'_{J,n_J})^{r'_{J,1}} || SID'_J)$, $R'_{J,1} = H((z'_{J,n_J+1})^{r'_{J,1}} || SID'_J)$, $X'_{J,1} = L'_{J,1} \oplus R'_{J,1}$ and $\widehat{X}'_J = \kappa'_{J,1} \oplus R'_{J,1}$. Next, S_J checks the XOR equation: $X'_{J,1} \oplus X'_{J,n_J+1} \oplus \dots \oplus X'_{J,n_J+m} \oplus X'_{J,n_J} = 0$. If the XOR equation is not correct, S_J will place the invalid XOR equation to a list BL_X and broadcast $\{S_J || BL_X || \bar{\sigma}'_{J,1}\}$, where $\bar{\sigma}'_{J,1} = \mathcal{S}(S_J || \tau'_J || BL_X || v)$. Otherwise, if the XOR equation is correct, then S_J broadcast $\{S_J || \widehat{X}'_J || X'_{J,1} || \bar{\sigma}'_{J,1}\}$, where $\bar{\sigma}'_{J,1} = \mathcal{S}(S_J || \tau'_J || \widehat{X}'_J || X'_{J,1} || v)$.

Round 6 :

Case 1 - $U_{J,k}$ and U_{J,n_J} receives $\{S_J || \widehat{X}'_J || X'_{J,1} || \bar{\sigma}'_{J,1}\}$: If $U_{J,k}$ for $n_J + 1 \leq k \leq n_J + m$ and U_{J,n_J} receives $\{S_J || \widehat{X}'_J || X'_{J,1} || \bar{\sigma}'_{J,1}\}$ from S_J , it means that S_J has verified that all signatures $\bar{\sigma}'_{J,i}$ for $n_J + 1 \leq i \leq n_J + m$ and $\bar{\sigma}'_{J,n_J}$ are valid and the XOR equation is correct in Round 5. In this case, $U_{J,k}$ and U_{J,n_J} first verify received signature $\bar{\sigma}'_{J,1}$. If the verification is not correct, then $U_{J,k}$ and U_{J,n_J} rejects S_J . Otherwise, $U_{J,k}$ and U_{J,n_J} computes

$$\begin{aligned} \overline{R}'_{J,k+1} &= X'_{J,k+1} \oplus R'_{J,k} \\ \overline{R}'_{J,k+2} &= X'_{J,k+2} \oplus \overline{R}'_{J,k+1} \\ &\vdots \\ \overline{R}'_{J,k-1} &= X'_{J,k-1} \oplus \overline{R}'_{J,k-2} \end{aligned}$$

and compares $\overline{R}'_{J,k-1}$ with $L'_{J,k}$. If $\overline{R}'_{J,k-1} = L'_{J,k}$, then all $U_{J,k}$ and U_{J,n_J} have $R'_{J,1}$ and uses $R'_{J,1}$ to obtain $\overline{\kappa}'_{J,1}$ from $\widehat{X}'_J$. Next, $U_{J,k}$ and U_{J,n_J} checks if $H(\overline{\kappa}'_{J,1} || v) = \tau'_J$ holds. If yes, all $U_{J,k}$, U_{J,n_J} and S_J computes the new subgroup key K'_J as follows:

$$K'_J = H(\kappa'_{J,1} || \kappa'_{J,n_J+1} || \dots || \kappa'_{J,n_J+m} || \kappa'_{J,n_J} || v) \quad (9.2)$$

Case 2 - $U_{J,k}$ and U_{J,n_J} receives $\{S_J || BL_X || \bar{\sigma}'_{J,1}\}$: If $U_{J,k}$ for $n_J + 1 \leq k \leq n_J + m$ and U_{J,n_J} receives $\{S_J || BL_X || \bar{\sigma}'_{J,1}\}$ from S_J , it means that S_J has detected either invalid signatures or the XOR equation is not correct in Round 5. In this case, $U_{J,k}$ and U_{J,n_J} first verify received signature $\bar{\sigma}'_{J,1}$. If the verification is not correct, then $U_{J,k}$ and U_{J,n_J} rejects S_J . Otherwise, $U_{J,k}$ and U_{J,n_J} examines the list BL_X to determine if S_J detected invalid signatures or invalid XOR equation in Round 5.

- [Invalid Signatures] If S_J detected invalid signatures in Round 5, then list BL_X contains the identities of invalid users $\tilde{U}_{J,k}$ that sent the invalid signature $\bar{\sigma}'_{J,k}$. Each $U_{J,k}$ and U_{J,n_J} then discards the invalid users $\tilde{U}_{J,k}$ from the GKA.
- [Invalid XOR Equation] If S_J detected an invalid XOR equation, then list BL_X contains the invalid XOR equation instead of invalid users' identities. The detection protocol described in Section 9.3 will be used to detect and block malicious group members $\tilde{U}_{J,k}$ that sent wrong information to disrupt the GKA.

For brevity, we shall use the term $\tilde{U}_{J,k}$ to denote invalid users and malicious group members. After detecting all invalid users and malicious group members $\tilde{U}_{J,k}$, the honest left neighbor $U_{J,k-1}$, which has not been blocked, of $\tilde{U}_{J,k}$ will re-compute $L'_{J,k-1} = H((z'_{J,k-2})^{r'_{J,k-1}} || \widetilde{SID'_J})$, $R'_{J,k-1} = H((z'_{J,k+1})^{r'_{J,k-1}} || \widetilde{SID'_J})$, $X'_{J,k-1} = L'_{J,k-1} \oplus R'_{J,k-1}$ and transmit $X'_{J,k-1}$ with signature, where $\widetilde{SID'_J}$ is equal to SID'_J excluding the identities of the discarded users in BL_X . The honest right neighbor $U_{J,k+1}$, which has not been blocked, of $\tilde{U}_{J,k}$ will re-compute $L'_{J,k+1} = H((z'_{J,k-1})^{r'_{J,k+1}} || \widetilde{SID'_J})$, $R'_{J,k+1} = H((z'_{J,k+2})^{r'_{J,k+1}} || \widetilde{SID'_J})$, $X'_{J,k+1} = L'_{J,k+1} \oplus R'_{J,k+1}$ and transmit $X'_{J,k+1}$ with signature. It should be noted that if left neighbor $U_{J,k-1}$ of $\tilde{U}_{J,k}$ is also an invalid user or malicious group member that has been blocked, then we set $U_{J,k-2} = U_{J,k-1}$. Similarly, if right neighbor $U_{J,k+1}$ of $\tilde{U}_{J,k}$ is also an invalid user or malicious group member that has been blocked, then we set $U_{J,k+2} = U_{J,k+1}$. All other users $U_{J,k}$ will use the new session identity $\widetilde{SID'_J}$ to re-compute $L'_{J,k}$, $R'_{J,k}$, $X'_{J,k}$ as in Round 4 and transmit $X'_{J,k}$ with signature. Next, Round 5 and Round 6 can continue without invalid signatures and invalid XOR equations detected in Round 5 as all the invalid users and malicious group members $\tilde{U}_{J,k}$ have been detected and blocked.

Phase 2: Final Group Key Generation

Round 1 : S_J computes $y'_J = g^{K'_J} \bmod p$ and sends $\{S_J || y'_J || \sigma'_J\}$ to main controller C , where $\sigma'_J = \mathcal{S}(S_J || y'_J)$.

Round 2 : Main controller C first verifies the received signature σ'_J . If the verification is not correct, then C rejects S_J . Otherwise, C selects random $r'_C \in Z_q^*$ and computes $y'_C = g^{r'_C} \bmod p$. Next, C computes the new DH key $T'_J = (y'_J)^{r'_C} \bmod p$ and $T'_i = y_i^{r'_C} \bmod p$ that it shares with S_J and S_i for $1 \leq i \neq J \leq n_c$ respectively. C then computes the final group key $K' = (y'_C \cdot y'_J \cdot \prod_{i=1, i \neq J}^{n_c} y_i)^{r'_C}$ and the key confirmation code $\theta'_C = H(K' || y'_C)$. Next, C encrypts the final group key K' using new DH key $T'_J = g^{r'_C K'_J}$ and $T'_i = g^{r'_C K_i}$ for $1 \leq i \neq J \leq n_c$ and obtains $c'_J = \mathcal{E}_{T'_J}(K')$ and $c'_i = \mathcal{E}_{T'_i}(K')$ for $1 \leq i \neq J \leq n_c$ respectively. Main controller C then broadcasts $\{C || y'_C || c'_J || c'_i || \theta'_C || \sigma'_C\}$ for $1 \leq i \neq J \leq n_c$, where $\sigma'_C = \mathcal{S}(C || y'_C || \theta'_C)$.

Phase 3: Distributing the Final Group Key

Round 1 :

(i) S_J first verifies signature σ'_C . If the verification is not correct, then S_J rejects C . Otherwise, S_J computes new DH key $T'_J = (y'_C)^{K'_J} \bmod p$ shared with C . Next, S_J uses T'_J to decrypt c'_J and obtains the new final group key K' . S_J then confirms and accepts the new final group key K' if $H(K' || y'_C) = \theta'_C$. Next, S_J encrypts $K' || S_J$ using new subgroup key K'_J and $K'_J || K' || S_J$ using previous subgroup key K_J and obtains $\bar{c}'_J = \mathcal{E}_{K'_J}(K' || S_J)$ and $\bar{c}''_J = \mathcal{E}_{K_J}(K'_J || K' || S_J)$ respectively. Finally, S_J computes key confirmation code $\theta'_J = H(K'_J || K' || \tau'_J)$ and broadcasts $\{S_J || \bar{c}'_J || \bar{c}''_J || \theta'_J || \bar{\sigma}'_J\}$ where $\bar{\sigma}'_J = \mathcal{S}(S_J || \tau'_J || \theta'_J || v)$ to subgroup members in $\mathcal{G}'_J$.

(ii) All other S_i for $1 \leq i \neq J \leq n_c$ first verifies signature σ'_C . If the verification is not correct, S_i rejects C . Otherwise, S_i computes new DH key $T'_i = (y'_C)^{K_i} \bmod p$ and uses T'_i to decrypt c'_i and obtains the new final group key K' . S_i then confirms and accepts the new final group key K' if $H(K' || y'_C) = \theta'_C$. Next, S_i encrypts $K' || S_i$ using subgroup key K_i and obtains $\bar{c}'_i = \mathcal{E}_{K_i}(K' || S_i)$. Finally, S_i computes the key confirmation code $\theta'_i = H(K_i || K' || \tau_i)$ and broadcasts $\{S_i || \bar{c}'_i || \theta'_i || \bar{\sigma}'_i\}$ where $\bar{\sigma}'_i = \mathcal{S}(S_i || \tau_i || \theta'_i || v)$ to subgroup members in $\mathcal{G}_i$.

Round 2 :

(i) Each $U_{J,k}$ for $n_J + 1 \leq k \leq n_J + m$ and U_{J,n_J} first verifies the signature $\bar{\sigma}'_J$. If the verification is not correct, then $U_{J,k}$ and U_{J,n_J} rejects S_J . Otherwise, $U_{J,k}$ and U_{J,n_J} uses new subgroup key K'_J to decrypt $\bar{c}'_J$ and obtains the identity S_J and final group key K' . If the identity S_J was not decrypted

correctly, then $\mathcal{U}_{J,k}$ and $\mathcal{U}_{J,n_J}$ rejects $\bar{c}'_J$. Otherwise, $\mathcal{U}_{J,k}$ and $\mathcal{U}_{J,n_J}$ checks if $\theta'_J = H(K'_J || K' || \tau'_J)$. If they are equal, $\mathcal{U}_{J,k}$ and $\mathcal{U}_{J,n_J}$ accepts the final group key K' and both K'_J and K' are confirmed.

(ii) Each $\mathcal{U}_{J,j}$ for $2 \leq j \leq n_J - 1$ first verifies the signature $\bar{\sigma}'_J$. If the verification is not correct, $\mathcal{U}_{J,j}$ rejects $\mathbf{S}_J$. Otherwise, $\mathcal{U}_{J,j}$ uses previous subgroup key K_J to decrypt $\bar{c}''_J$ and obtains the identity $\mathbf{S}_J$, new subgroup key K'_J and new final group key K' . If the identity $\mathbf{S}_J$ was not decrypted correctly, then $\mathcal{U}_{J,j}$ rejects $\bar{c}''_J$. Otherwise, $\mathcal{U}_{J,j}$ checks if $\theta'_J = H(K'_J || K' || \tau'_J)$. If they are equal, $\mathcal{U}_{J,j}$ accepts the new subgroup key K'_J and new final group key K' and both K'_J and K' are confirmed.

(iii) Each $\mathcal{U}_{i,j}$ for $1 \leq i \neq J \leq n_c$ and $2 \leq j \leq n_i$ first verifies the signature $\bar{\sigma}'_i$. If the verification is not correct, then $\mathcal{U}_{i,j}$ rejects $\mathbf{S}_i$. Otherwise, $\mathcal{U}_{i,j}$ uses subgroup key K_i to decrypt $\bar{c}'_i$ and obtains the identity $\mathbf{S}_i$ and final group key K' . If the identity $\mathbf{S}_i$ was not decrypted correctly, then $\mathcal{U}_{i,j}$ rejects $\bar{c}'_i$. Otherwise, $\mathcal{U}_{i,j}$ checks if $\theta'_i = H(K_i || K' || \tau_i)$. If they are equal, $\mathcal{U}_{i,j}$ accepts and confirms the new final group key K' .

9.2.3 Leave Protocol

Suppose a set of ℓ users $\mathcal{L} = \{\mathcal{U}_{L,l_1}, \dots, \mathcal{U}_{L,l_\ell}\}$ wishes to leave subgroup $\mathcal{G}_L = \{\mathcal{U}_{L,1}, \dots, \mathcal{U}_{L,n_L}\}$, then the Leave Protocol will be executed to establish a new subgroup key K'_L and new final group key K' to protect future group communications from the leaving users. For any leaving user $\mathcal{U}_{L,l_i}$, the left neighbor is denoted as $\mathcal{U}_{L,l_i-w_i}$ if the w_i consecutive users $\mathcal{U}_{L,l_i}, \mathcal{U}_{L,l_i-1}, \dots, \mathcal{U}_{L,l_i-(w_i-1)}$ are all leaving and $\mathcal{U}_{L,l_i-w_i}$ is not leaving the group. Similarly, the right neighbor is denoted as $\mathcal{U}_{L,l_i+w'_i}$ if the w'_i consecutive users $\mathcal{U}_{L,l_i}, \mathcal{U}_{L,l_i+1}, \dots, \mathcal{U}_{L,l_i+(w'_i-1)}$ are all leaving and $\mathcal{U}_{L,l_i+w'_i}$ is not leaving the group. We let $\mathcal{N}(\mathcal{R})$ be the set of all left and right neighbors of leaving users, i.e. $\mathcal{N}(\mathcal{R}) = \{\mathcal{U}_{L,l_1-w_1}, \mathcal{U}_{L,l_1+w'_1}, \dots, \mathcal{U}_{L,l_\ell-w_\ell}, \mathcal{U}_{L,l_\ell+w'_\ell}\}$, and re-index the remaining users in subgroup $\mathcal{G}'_L = \mathcal{G}_L \setminus \mathcal{L} = \{\mathcal{V}_{L,1}, \dots, \mathcal{V}_{L,n_L-\ell}\}$. We also let $\mathcal{V}_{L,k} \in \mathcal{N}(\mathcal{R})$ denote the re-indexed remaining users that are left and right neighbors of leaving users.

The Leave Protocol consists of 3 phases. During Phase 1, subgroup controller $\mathbf{S}_L = \mathcal{V}_{L,1}$ of subgroup $\mathcal{G}'_L$ will interact with remaining users $\mathcal{V}_{L,j}$ for $2 \leq j \leq n_L - \ell$ and compute the new subgroup key K'_L . Next in Phase 2, $\mathbf{S}_L$ uses K'_L to interact with main controller $\mathbf{C}$ for $\mathbf{C}$ to compute the new final group key K' . Finally in Phase 3, the new final group key K' is disseminated from $\mathbf{C}$ to all user nodes via subgroup controllers $\mathbf{S}_i$ for $1 \leq i \leq n_c$ (including $\mathbf{S}_L$). $\mathbf{S}_L$ will use the

new subgroup key K'_L to encrypt and send the new final group key K' to remaining users $V_{L,j}$ for $2 \leq j \leq n_L - \ell$. Key confirmation messages are also sent to verify and confirm the new subgroup key K'_L and the new final group key K' . The Leave Protocol works as follows:

Phase 1: Subgroup Key Generation

Round 1 : Subgroup controller $S_L = V_{L,1}$ randomly selects $\kappa'_{L,1} \in \{0, 1\}^\lambda$, computes $\tau'_L = H(\kappa'_{L,1} || v)$ and broadcasts $\{S_L || \tau'_L || v\}$ to remaining subgroup members $V_{L,j}$ for $2 \leq j \leq n_L - \ell$, where λ is an integer and v is a counter that starts from 0 in the Initial GKA Protocol and increments by 1 whenever a dynamic GKA protocol such as the Join or Leave Protocol is executed.

Round 2 : Each remaining user $V_{L,k} \in \mathcal{N}(\mathcal{R})$, who is a left or right neighbor of leaving users, randomly selects $r'_{L,k} \in Z_q^*$ and computes $z'_{L,k} = g^{r'_{L,k}} \bmod p$. Next, $V_{L,k}$ broadcasts $\{V_{L,k} || z'_{L,k} || \sigma'_{L,k}\}$, where $\sigma'_{L,k} = \mathcal{S}(V_{L,k} || z'_{L,k} || \tau'_L || v)$ and stores τ'_L .

Round 3 : S_L checks the received signatures $\sigma'_{L,k}$ sent by $V_{L,k} \in \mathcal{N}(\mathcal{R})$ in Round 2. If S_L detects invalid signatures sent by certain malicious nodes, then these malicious nodes will be placed into a list BL_Z . If $S_L \in \mathcal{N}(\mathcal{R})$, then S_L selects random $r'_{L,1} \in Z_q^*$ and computes $z'_{L,1} = g^{r'_{L,1}} \bmod p$. Otherwise, if $S_L \notin \mathcal{N}(\mathcal{R})$, then S_L uses $z'_{L,1} = z_{L,1} = g^{r_{L,1}} \bmod p$ from the previous group key agreement. Finally, S_L broadcasts $\{S_L || z'_{L,1} || BL_Z || \sigma'_{L,1}\}$, where $\sigma'_{L,1} = \mathcal{S}(S_L || \tau'_L || z'_{L,1} || BL_Z || v)$.

Round 4 : Each $V_{L,j}$ for $2 \leq j \leq n_L - \ell$ first checks signature $\sigma'_{L,1}$. If the verification is not correct, then $V_{L,j}$ rejects S_L . Otherwise, $V_{L,j}$ checks the list BL_Z to identify the malicious nodes to be blocked from the GKA using the similar method described in the Initial Group Key Agreement and Join Protocol. Next, each $V_{L,j}$ computes $L'_{L,j} = H((z'_{L,j-1})^{r'_{L,j}} || SID'_L)$, $R'_{L,j} = H((z'_{L,j+1})^{r'_{L,j}} || SID'_L)$, where $z'_{L,n_L-\ell+1} = z'_{L,1}$ and $SID'_L = H(\tau'_L || V_{L,1} || V_{L,2} || \dots || V_{L,n_L-\ell})$. If $V_{L,j} \notin \mathcal{N}(\mathcal{R})$, then $r'_{L,j} = r_{L,j}$ and $z'_{L,j} = z_{L,j}$, where $r_{L,j}$ and $z_{L,j}$ are from previous GKA. Next, $V_{L,j}$ generates $X'_{L,j} = L'_{L,j} \oplus R'_{L,j}$. Finally, $V_{L,j}$ randomly selects $\kappa'_{L,j} \in \{0, 1\}^\lambda$ and broadcasts $\{V_{L,j} || \kappa'_{L,j} || X'_{L,j} || \bar{\sigma}'_{L,j}\}$, where $\bar{\sigma}'_{L,j} = \mathcal{S}(V_{L,j} || \kappa'_{L,j} || X'_{L,j} || \tau'_L || v)$.

Round 5 : S_L first checks the received signatures $\bar{\sigma}'_{L,j}$ for $2 \leq j \leq n_L - \ell$. If there are invalid signatures, S_L will place the identities of invalid users $\tilde{V}_{L,j}$ that generated the invalid signature to a list BL_X and broadcast $\{S_L || BL_X || \bar{\sigma}'_{L,1}\}$, where $\bar{\sigma}'_{L,1} = \mathcal{S}(S_L || \tau'_L || BL_X || v)$. Otherwise, if the signature verifications are correct, then S_L computes $L'_{L,1} = H((z'_{L,n_L-\ell})^{r'_{L,1}} || SID'_L)$ and $R'_{L,1} = H((z'_{L,2})^{r'_{L,1}} || SID'_L)$ where $r'_{L,1} = r_{L,1}$ if $S_L \notin \mathcal{N}(\mathcal{R})$, $z'_{L,n_L-\ell} = z_{L,n_L-\ell}$ if $V_{L,n_L-\ell} \notin \mathcal{N}(\mathcal{R})$ and

$z'_{L,2} = z_{L,2}$ if $V_{L,2} \notin \mathcal{N}(\mathcal{R})$. Next, S_L computes $X'_{L,1} = L'_{L,1} \oplus R'_{L,1}$ and $\widehat{X}'_L = \kappa'_{L,1} \oplus R'_{L,1}$. S_L then checks the XOR equation: $X'_{L,1} \oplus X'_{L,2} \oplus \dots \oplus X'_{L,n_L-\ell} = 0$. If the XOR equation is not correct, S_L will place the invalid XOR equation to list BL_x and broadcast $\{S_L || \text{BL}_x || \bar{\sigma}'_{L,1}\}$, where $\bar{\sigma}'_{L,1} = \mathcal{S}(S_L || \tau'_L || \text{BL}_x || v)$. Otherwise, if the XOR equation is correct, then S_L broadcast $\{S_L || \widehat{X}'_L || X'_{L,1} || \bar{\sigma}'_{L,1}\}$, where $\bar{\sigma}'_{L,1} = \mathcal{S}(S_L || \tau'_L || \widehat{X}'_L || X'_{L,1} || v)$.

Round 6 :

Case 1 - $V_{L,j}$ receives $\{S_L || \widehat{X}'_L || X'_{L,1} || \bar{\sigma}'_{L,1}\}$: If $V_{L,j}$ for $2 \leq j \leq n_L - \ell$ receives $\{S_L || \widehat{X}'_L || X'_{L,1} || \bar{\sigma}'_{L,1}\}$ from S_L , it means that S_L has verified that all signatures $\bar{\sigma}'_{L,k}$ for $2 \leq k \leq n_L - \ell$ are valid and the XOR equation is correct in Round 5. In this case, $V_{L,j}$ first verifies the received signature $\bar{\sigma}'_{L,1}$. If the verification is not correct, then $V_{L,j}$ rejects S_L . Otherwise, $V_{L,j}$ computes

$$\begin{aligned} \bar{R}'_{L,j+1} &= X'_{L,j+1} \oplus R'_{L,j} \\ \bar{R}'_{L,j+2} &= X'_{L,j+2} \oplus \bar{R}'_{L,j+1} \\ &\vdots \\ \bar{R}'_{L,j-1} &= X'_{L,j-1} \oplus \bar{R}'_{L,j-2} \end{aligned}$$

and compares $\bar{R}'_{L,j-1}$ with $L'_{L,j}$. If $\bar{R}'_{L,j-1} = L'_{L,j}$, then all $V_{L,j}$ have $R'_{L,1}$ and uses $R'_{L,1}$ to obtain $\bar{\kappa}'_{L,1}$ from $\widehat{X}'_L$. Next, $V_{L,j}$ checks if $H(\bar{\kappa}'_{L,1} || v) = \tau'_L$ holds. If yes, all $V_{L,j}$ and S_L computes the subgroup key K'_L as follows:

$$K'_L = H(\kappa'_{L,1} || \kappa'_{L,2} || \dots || \kappa'_{L,n_L-\ell} || v) \quad (9.3)$$

Case 2 - $V_{L,j}$ receives $\{S_L || \text{BL}_x || \bar{\sigma}'_{L,1}\}$: If $V_{L,j}$ for $2 \leq j \leq n_L - \ell$ receives $\{S_L || \text{BL}_x || \bar{\sigma}'_{L,1}\}$ from S_L , it means that S_L has detected either invalid signatures or invalid XOR equation in Round 5. In this case, $V_{L,j}$ first verifies the received signature $\bar{\sigma}'_{L,1}$. If the verification is not correct, $V_{L,j}$ rejects S_L . Otherwise, $V_{L,j}$ examines the list BL_x to determine if S_L detected invalid signatures or invalid XOR equation in Round 5.

- [Invalid Signatures] If S_L detected invalid signatures in Round 5, then list BL_x contains the identities of invalid users $\tilde{V}_{L,j}$ for $2 \leq j \leq n_L - \ell$ that sent the invalid signature $\bar{\sigma}'_{L,j}$. Each

$V_{L,j}$ then discards the invalid users $\tilde{V}_{L,j}$ from the GKA.

- [Invalid XOR Equation] If S_L detected an invalid XOR equation, then list BL_X contains the invalid XOR equation instead of invalid users' identities. The detection protocol described in Section 9.3 will be used to detect and block malicious group members $\tilde{V}_{L,j}$ that sent wrong information to disrupt the GKA.

For brevity, we shall use the term $\tilde{V}_{L,j}$ to denote invalid users and malicious group members. After detecting all invalid users and malicious group members $\tilde{V}_{L,j}$, the honest left neighbor $V_{L,j-1}$, which has not been blocked, of $\tilde{V}_{L,j}$ will re-compute $L'_{L,j-1} = H((z'_{L,j-2})^{r'_{L,j-1}} || \widetilde{SID'_L})$, $R'_{L,j-1} = H((z'_{L,j+1})^{r'_{L,j-1}} || \widetilde{SID'_L})$, $X'_{L,j-1} = L'_{L,j-1} \oplus R'_{L,j-1}$ and transmit $X'_{L,j-1}$ with signature, where $\widetilde{SID'_L}$ is equal to SID'_L excluding the identities of the discarded users in BL_X . The honest right neighbor $V_{L,j+1}$, which has not been blocked, of $\tilde{V}_{L,j}$ will re-compute $L'_{L,j+1} = H((z'_{L,j-1})^{r'_{L,j+1}} || \widetilde{SID'_L})$, $R'_{L,j+1} = H((z'_{L,j+2})^{r'_{L,j+1}} || \widetilde{SID'_L})$, $X'_{L,j+1} = L'_{L,j+1} \oplus R'_{L,j+1}$ and transmit $X'_{L,j+1}$ with signature. It should be noted that if left neighbor $V_{L,j-1}$ of $\tilde{V}_{L,j}$ is also an invalid user or malicious group member that has been blocked, then we set $V_{L,j-2} = V_{L,j-1}$. Similarly, if right neighbor $V_{L,j+1}$ of $\tilde{V}_{L,j}$ is also an invalid user or malicious group member that has been blocked, then we set $V_{L,j+2} = V_{L,j+1}$. All other users $V_{L,j}$ will use the new session identity $\widetilde{SID'_L}$ to re-compute $L'_{L,j}$, $R'_{L,j}$, $X'_{L,j}$ as in Round 4 and transmit $X'_{L,j}$ with signature. Next, Round 5 can continue without invalid signatures and invalid XOR equations detected as all the invalid users and malicious group members $\tilde{V}_{L,j}$ have been detected and blocked.

Phase 2: Final Group Key Generation

Round 1 : S_L computes $y'_L = g^{K'_L} \bmod p$ and sends $\{S_L || y'_L || \sigma'_L\}$ to main controller C , where $\sigma'_L = \mathcal{S}(S_L || y'_L)$.

Round 2 : Main controller C first verifies the received signature σ'_L . If the verification is not correct, then C rejects S_L . Otherwise, C selects random $r'_C \in Z_q^*$ and computes $y'_C = g^{r'_C} \bmod p$. Next, C computes the new DH key $T'_L = (y'_L)^{r'_C} \bmod p$ and $T'_i = y_i^{r'_C} \bmod p$ that it shares with S_L and S_i for $1 \leq i \neq L \leq n_c$ respectively. C then computes the final group key $K' = (y'_C \cdot y'_L \cdot \prod_{i=1, i \neq L}^{n_c} y_i)^{r'_C}$ and the key confirmation code $\theta'_C = H(K' || y'_C)$. Next, C encrypts the final group key K' using the new DH key $T'_L = g^{r'_C K'_L}$ and $T'_i = g^{r'_C K_i}$ for $1 \leq i \neq L \leq n_c$ and obtains $c'_L = \mathcal{E}_{T'_L}(K')$ and $c'_i = \mathcal{E}_{T'_i}(K')$

for $1 \leq i \neq L \leq n_c$ respectively. Main controller $\mathbf{C}$ then broadcasts $\{\mathbf{C}||y'_c||c'_L||c'_i||\theta'_c||\sigma'_c\}$ for $1 \leq i \neq L \leq n_c$, where $\sigma'_c = \mathcal{S}(\mathbf{C}||y'_c||\theta'_c)$.

Phase 3: Distributing the Final Group Key

Round 1 :

(i) $\mathbf{S}_L$ first verifies signature σ'_c . If the verification is not correct, then $\mathbf{S}_L$ rejects $\mathbf{C}$. Otherwise, $\mathbf{S}_L$ computes new DH key $T'_L = (y'_c)^{K'_L} \bmod p$ shared with $\mathbf{C}$. Next, $\mathbf{S}_L$ uses T'_L to decrypt c'_L and obtain the new final group key K' . $\mathbf{S}_L$ then confirms and accepts the final group key K' if $H(K'||y'_c) = \theta'_c$. Next, $\mathbf{S}_L$ encrypts $K'||\mathbf{S}_L$ using new subgroup key K'_L and obtains $\bar{c}'_L = \mathcal{E}_{K'_L}(K'||\mathbf{S}_L)$. Finally, $\mathbf{S}_L$ computes key confirmation code $\theta'_L = H(K'_L||K'||\tau'_L)$ and broadcasts $\{\mathbf{S}_L||\bar{c}'_L||\theta'_L||\bar{\sigma}'_L\}$ where $\bar{\sigma}'_L = \mathcal{S}(\mathbf{S}_L||\tau'_L||\theta'_L||v)$ to subgroup members $\mathbf{V}_{L,j}$ for $2 \leq j \leq n_L - \ell$.

(ii) All other $\mathbf{S}_i$ for $1 \leq i \neq L \leq n_c$ first verifies signature σ'_c . If the verification is not correct, then $\mathbf{S}_i$ rejects $\mathbf{C}$. Otherwise, $\mathbf{S}_i$ computes new DH key $T'_i = (y'_c)^{K_i} \bmod p$ and uses T'_i to decrypt c'_i and obtains the final group key K' . $\mathbf{S}_i$ then confirms and accepts the final group key K' if $H(K'||y'_c) = \theta'_c$. Next, $\mathbf{S}_i$ encrypts $K'||\mathbf{S}_i$ using subgroup key K_i and obtains $\bar{c}'_i = \mathcal{E}_{K_i}(K'||\mathbf{S}_i)$. Finally, $\mathbf{S}_i$ computes the key confirmation code $\theta'_i = H(K_i||K'||\tau_i)$ and broadcasts $\{\mathbf{S}_i||\bar{c}'_i||\theta'_i||\bar{\sigma}'_i\}$ where $\bar{\sigma}'_i = \mathcal{S}(\mathbf{S}_i||\tau_i||\theta'_i||v)$ to subgroup members $\mathbf{U}_{i,j}$ for $2 \leq j \leq n_i$.

Round 2 : Each $\mathbf{V}_{L,j}$ ($\mathbf{U}_{i,j}$) for $2 \leq j \leq n_L - \ell$ ($1 \leq i \neq L \leq n_c, 2 \leq j \leq n_i$) verifies the signature $\bar{\sigma}'_L$ ($\bar{\sigma}'_i$). If the signature verification of $\bar{\sigma}'_L$ ($\bar{\sigma}'_i$) is not correct, then $\mathbf{V}_{L,j}$ ($\mathbf{U}_{i,j}$) rejects $\mathbf{S}_L$ ($\mathbf{S}_i$). Otherwise, $\mathbf{V}_{L,j}$ ($\mathbf{U}_{i,j}$) uses subgroup key K'_L (K_i) to decrypt $\bar{c}'_L$ ($\bar{c}'_i$) and obtains the identity $\mathbf{S}_L$ ($\mathbf{S}_i$) and final group key K' . If the identity $\mathbf{S}_L$ ($\mathbf{S}_i$) was not decrypted correctly, then $\mathbf{V}_{L,j}$ ($\mathbf{U}_{i,j}$) rejects $\bar{c}'_L$ ($\bar{c}'_i$). Otherwise, $\mathbf{V}_{L,j}$ checks if $\theta'_L = H(K'_L||K'||\tau'_L)$. If they are equal, $\mathbf{V}_{L,j}$ accepts the final group key K' and both K'_L and K' are confirmed. Similarly, $\mathbf{U}_{i,j}$ checks if $\theta'_i = H(K_i||K'||\tau_i)$. If they are equal, $\mathbf{U}_{i,j}$ accepts and confirms the new final group key K' .

9.3 Detection of Malicious Group Members

In this section, we describe the proposed detection protocol that will be executed in Phase 1 Round 5 of the proposed Initial GKA, Join and Leave Protocol if an invalid XOR equation is detected. The detection protocol can be used to detect malicious group members that try to disrupt GKA services

by sending “wrong” or “bad” messages. After detecting and blocking malicious group members, the GKA protocol will continue to establish subgroup keys and final group key, thus preventing insider attacks. It should be noted that we assume the malicious group members are independent and do not collaborate with each other.

The detection protocol consists of 2 rounds during which the users $U_{a,j}$ for $2 \leq j \leq n_a$ of subgroup $\mathcal{G}_a$ with the invalid XOR equation will send their left and right keys, $L_{a,j}$ and $R_{a,j}$ respectively, to the subgroup controller S_a securely, where n_a is the number of members in subgroup $\mathcal{G}_a$. Next in Round 2, S_a will check and determine the malicious group members that will be blocked from GKA and send a list containing these blocked malicious group members to the remaining honest users. The original GKA protocol can then continue to establish the subgroup keys and final group key without interference from malicious group members. The detection protocol is as follows:

Round 1 : Each user $U_{a,j} \in \mathcal{G}_a$ for $2 \leq j \leq n_a$ first computes a DH key $T_{a,j} = z_{a,1}^{r_{a,j}} \bmod p$ that it shares with subgroup controller S_a . $U_{a,j}$ then encrypts right key $R_{a,j}$, left key $L_{a,j}$ and its identity $U_{a,j}$ using $T_{a,j}$ and obtains $c_{a,j} = \mathcal{E}_{T_{a,j}}(R_{a,j}||L_{a,j}||U_{a,j})$. Finally, $U_{a,j}$ sends $\{U_{a,j}||c_{a,j}||\sigma_{a,j}\}$ to S_a , where $\sigma_{a,j} = \mathcal{S}(U_{a,j}||\tau_a||c_{a,j})$.

Round 2 : Subgroup controller S_a first verifies all the received signatures $\sigma_{a,j}$ for $2 \leq j \leq n_a$. If the verifications are not correct, then S_a rejects $U_{a,j}$. Otherwise, S_a computes $T_{a,j} = z_{a,1}^{r_{a,j}} \bmod p$ and decrypts $c_{a,j}$ and obtains right key $R_{a,j}$, left key $L_{a,j}$ and identity $U_{a,j}$ respectively for $2 \leq j \leq n_a$ and checks if identity $U_{a,j}$ was decrypted correctly. If yes, S_a checks the equation $X_{a,j} = L_{a,j+1} \oplus R_{a,j-1}$ for $2 \leq j \leq n_a$. Suppose $U_{a,k}$ (for some k) is the malicious group member that sent the incorrect $X_{a,k}$ that cause the invalid XOR equation in Phase 1 Round 5, then S_a will detect that $X_{a,k} \neq L_{a,k+1} \oplus R_{a,k-1}$. Next, S_a will place the identity $U_{a,k}$ into a list BL_D . In this case, it means that either the left key $L_{a,k}$ is incorrect or right key $R_{a,k}$ is incorrect or both are incorrect. To determine if left key $L_{a,k}$ is incorrect, S_a checks the following equations: $X_{a,k} \oplus L_{a,k+1} = R_{a,k-1}$ and $R_{a,k-1} = L_{a,k}$. If either equation is incorrect, it means that left key $L_{a,k}$ is incorrect. Similarly, to determine if right key $R_{a,k}$ is incorrect, S_a checks the following equations: $X_{a,k} \oplus R_{a,k-1} = L_{a,k+1}$ and $L_{a,k+1} = R_{a,k}$. If either check is incorrect, it means that right key $R_{a,k}$ is incorrect. Finally, S_a sends $\{S_a||BL_D||\sigma_a\}$ where $\sigma_a = \mathcal{S}(S_a||\tau_a||BL_D)$ to inform honest users of the malicious group

members to be blocked from GKA.

9.4 Security Analysis

In this section, we provide the security analysis for the three proposed protocols, namely the Initial GKA, Join and Leave Protocol. We show that the three proposed GKA protocols are secure as follows:

9.4.1 Security Analysis of Initial GKA Protocol

As Phase 1 of the proposed Initial GKA protocol (without the signature information sent) is similar to Kim's GKA protocol [KLL04], we call it a variant of Kim's GKA protocol (VK-GKA) and prove that the VK-GKA is secure as follows:

Theorem 27 *If Kim's GKA protocol is secure, then VK-GKA protocol is secure.*

Proof: : This theorem proceeds by reduction proof, that is, for any instance of VK-GKA, it is reduced to an instance of Kim's GKA protocol. Now, consider a VK-GKA protocol as follows:

Round 1: The first user U_1 chooses a random $\kappa_1 \in \{0, 1\}^\lambda$, computes $\tau = H(\kappa_1 || 0)$ and sends τ .

Round 2: Each user U_i for $2 \leq i \leq n$ chooses a random $r_i \in Z_p^*$, computes $z_i = g^{r_i}$ and sends z_i .

Round 3: The first user U_1 chooses a random $r_1 \in Z_p^*$, computes $z_1 = g^{r_1}$ and sends z_1 .

Round 4: Upon receipt of all z_j for $1 \leq j \leq n$, each U_i for $2 \leq i \leq n$ computes $L_i = H(z_{i-1}^{r_i} || \mathbf{SID})$, $R_i = H(z_{i+1}^{r_i} || \mathbf{SID})$ and $X_i = L_i \oplus R_i$, where $\mathbf{SID} = H(\tau || U_1 || U_2 || \dots || U_n)$. U_i then chooses a random $\kappa_i \in \{0, 1\}^\lambda$ and sends out $\kappa_i || X_i$.

Round 5: Upon receipt of $\kappa_j || X_j$ for $2 \leq j \leq n$, U_1 computes $L_1 = H(z_n^{r_1} || \mathbf{SID})$, $R_1 = H(z_2^{r_1} || \mathbf{SID})$, $X_1 = L_1 \oplus R_1$ and $\hat{X} = \kappa_1 \oplus R_1$ and sends out $\hat{X} || X_1$.

Round 6: U_i for $2 \leq i \leq n$ computes

$$\bar{R}_{i+1} = X_{i+1} \oplus R_i, \quad \bar{R}_{i+2} = X_{i+2} \oplus \bar{R}_{i+1}, \quad \dots, \quad \bar{R}_{i-1} = X_{i-1} \oplus \bar{R}_{i-2}.$$

If $\bar{R}_{i-1} = L_i$, then U_i uses R_1 to obtain $\bar{\kappa}_1$ from $\hat{X}$. If $H(\bar{\kappa}_1||0) = \tau$ holds, then U_i for $1 \leq i \leq n$ computes the group key K as follows:

$$K = H(\kappa_1||\kappa_2||\cdots||\kappa_{n-1}||\kappa_n||0).$$

Now, we convert the above VK-GKA protocol into Kim's GKA protocol as follows: We combine Round 1, Round 2 and Round 3 of the VK-GKA protocol into Round 1' and we also combine Round 4 and Round 5 in the VK-GKA protocol into Round 2' as follows:

Round 1': The first user U_1 chooses a random $\kappa_1 \in \{0, 1\}^\lambda$ and $r_1 \in Z_p^*$, computes $z_1 = g^{r_1}$ and $\tau = H(\kappa_1||0)$ and sends $\tau||z_1$. Each user U_i for $2 \leq i \leq n$ chooses a random $r_i \in Z_p^*$, computes $z_i = g^{r_i}$ and sends z_i .

Round 2': Upon receipt of all z_j for $1 \leq j \leq n$, each U_i for $1 \leq i \leq n$ computes $L_i = H(z_{i-1}^{r_i}||SID)$, $R_i = H(z_{i+1}^{r_i}||SID)$ and $X_i = L_i \oplus R_i$, where $SID = H(\tau||U_1||U_2||\cdots||U_n)$. The first user U_1 in additionally computes $\hat{X} = \kappa_1 \oplus R_1$. U_i for $2 \leq i \leq n$ then chooses a random $\kappa_i \in \{0, 1\}^\lambda$ and sends out $\kappa_i||X_i$ and first user U_1 sends out $\hat{X}||X_1$.

Computation of Key: U_i for $1 \leq i \leq n$ computes

$$\bar{R}_{i+1} = X_{i+1} \oplus R_i, \quad \bar{R}_{i+2} = X_{i+2} \oplus \bar{R}_{i+1}, \quad \cdots, \quad \bar{R}_{i-1} = X_{i-1} \oplus \bar{R}_{i-2}.$$

If $\bar{R}_{i-1} = L_i$, then U_i uses R_1 to obtain $\bar{\kappa}_1$ from $\hat{X}$. If $H(\bar{\kappa}_1||0) = \tau$ holds, then U_i for $1 \leq i \leq n$ computes the group key K as follows:

$$K = H(\kappa_1||\kappa_2||\cdots||\kappa_{n-1}||\kappa_n||0).$$

Then, VK-GKA protocol reduces to a Kim's GKA protocol. Hence, VK-GKA protocol is secure if Kim's GKA protocol is secure. ■

As Phase 2 of the proposed Initial GKA protocol (without the signature information sent) is a variant of the NLKW protocol [NLKW05] with introducing a symmetric key encryption scheme SKE and the NLKW protocol is a STAR structure (where there is a main controller that communicates

with all the user nodes), we call the variant of the NLKW protocol as VSTAR protocol and describe as follows.

Round 1: Each user U_i ($1 \leq i \leq n$) chooses a random $r_i \in Z_q^*$, computes $z_i = g^{r_i}$ and sends z_i .

Round 2: Upon receipt of all z_i for $1 \leq i \leq n$, controller $\mathbf{C}$ chooses a random $r_c \in Z_q^*$, computes $z_c = g^{r_c}$, $T_i = z_i^{r_c}$ for $1 \leq i \leq n$ and group key $K = (z_c \cdot \prod_{i=1}^n z_i)^{r_c} = g^{r_c r_c + \sum_{i=1}^n r_c r_i}$. Next, $\mathbf{C}$ encrypts K as $c_i = \mathcal{E}_{T_i}(K)$ for $1 \leq i \leq n$ and sends out z_c and c_i for $1 \leq i \leq n$.

Computation of Key: Upon receipt of z_c and c_i , each user U_i ($1 \leq i \leq n$) computes $T_i = z_c^{r_i}$ and obtains the group key K by decrypting c_i .

We state the security of the VSTAR protocol as follows:

Theorem 28 *If CDH problem is difficult and the symmetric key encryption scheme SKE is secure, then the VSTAR protocol is secure.*

Proof: : Since g^{r_i} for $1 \leq i \leq n$ and g^{r_c} are sent in round 1 and round 2 respectively, therefore, g^{r_c} and $g^{r_c + r_1 + \dots + r_n}$ are public information. If an adversary can compute $K = g^{r_c r_c + \sum_{i=1}^n r_c r_i} = g^{r_c(r_c + r_1 + \dots + r_n)}$, then this contradicts to the assumption that the CDH problem is difficult as $(g^{r_c}, g^{r_c + r_1 + \dots + r_n})$ is a Diffie-Hellman pair. Furthermore, $T_i = g^{r_c r_i}$ is the key for encrypting K as $c_i = \mathcal{E}_{T_i}(K)$ for $1 \leq i \leq n$ and computing T_i is a CDH problem; and the symmetric key encryption scheme is secure. Therefore, it is not possible for an adversary to obtain the key K from c_i . Hence, we conclude that the VSTAR protocol is secure. ■

Next, we prove that the proposed Initial GKA Protocol is secure. We consider two cases, namely, the up flow information (Phase 1 and 2) and down flow information (Phase 3). The up flow information are the information sent in the various subgroups $\mathcal{G}_i$ for $1 \leq i \leq n_c$ and from subgroup controllers S_i to main controller $\mathbf{C}$ to form the various subgroup keys K_i and final group key K respectively. The down flow information is basically a dissemination of the final group key K from the main controller $\mathbf{C}$ to the user nodes $U_{i,j}$ for $2 \leq j \leq n_i$ via subgroup controllers S_i .

Theorem 29 *Given the up flow information (Phase 1 and 2), an adversary cannot obtain the subgroup keys K_i for $1 \leq i \leq n_c$ and the final group key K in the proposed Initial GKA protocol provided the VK-GKA protocol, the VSTAR protocol and the RSA signature scheme are secure.*

Proof: In this proof, we consider two cases, that is, (a) the adversary $\mathcal{A}$ tries to impersonate subgroup controller $\mathbf{S}_i = \mathbf{U}_{i,1}$ ($1 \leq i \leq n_c$), user $\mathbf{U}_{i,j}$ ($1 \leq i \leq n_c$ and $2 \leq j \leq n_i$) or main controller $\mathbf{C}$ to establish the subgroup keys K_i ($1 \leq i \leq n_c$) and the final group key K , and (b) the adversary $\mathcal{A}$ tries to obtain subgroup keys K_i ($1 \leq i \leq n_c$) and the final group key K from the up flow information sent in Phase 1 and 2. For case (a), we first consider the adversary $\mathcal{A}$ tries to impersonate subgroup controller $\mathbf{S}_i$. Then in Phase 1 Round 1, $\mathcal{A}$ generates $\tau_i = \mathbf{H}(\kappa_{i,1}||0)$. Next in Phase 1 Round 3, $\mathcal{A}$ first generates $z_{i,1} = g^{r_{i,1}} \bmod p$, then $\mathcal{A}$ needs to compute the valid signature $\sigma_{i,1} = \mathcal{S}(\mathbf{S}_i||\tau_i||z_{i,1}||\mathbf{BL}_Z||0)$. The validity of signature $\sigma_{i,1}$ is checked by $\mathbf{U}_{i,j}$ for $2 \leq j \leq n_i$ in Phase 1 Round 4. If the signature $\sigma_{i,1}$ is not correct, then $\mathbf{U}_{i,j}$ rejects $\mathbf{S}_i$. As $\mathcal{A}$ does not know the RSA private key d_i of $\mathbf{S}_i$ and the RSA signature scheme is assumed to be secure, therefore $\mathcal{A}$ is not able to produce a valid signature $\sigma_{i,1}$. Next suppose $\mathcal{A}$ tries to impersonate $\mathbf{S}_i$ in Phase 1 Round 5, then $\mathcal{A}$ first generates $X_{i,1} = L_{i,1} \oplus R_{i,1}$ and $\widehat{X}_i = \kappa_{i,1} \oplus R_{i,1}$ where $L_{i,1} = \mathbf{H}(z_{i,n_i}^{r_{i,1}}||\mathbf{SID}_i)$ and $R_{i,1} = \mathbf{H}(z_{i,2}^{r_{i,1}}||\mathbf{SID}_i)$. Then, $\mathcal{A}$ has to compute the valid signature $\bar{\sigma}_{i,1} = \mathcal{S}(\mathbf{S}_i||\tau_i||\widehat{X}_i||X_{i,1}||0)$. The validity of signature $\bar{\sigma}_{i,1}$ is checked by $\mathbf{U}_{i,j}$ for $2 \leq j \leq n_i$ in Phase 1 Round 6. If the signature $\bar{\sigma}_{i,1}$ is not correct, then $\mathbf{U}_{i,j}$ rejects $\mathbf{S}_i$. As $\mathcal{A}$ does not know the RSA private key d_i of $\mathbf{S}_i$ and the RSA signature scheme is assumed to be secure, therefore $\mathcal{A}$ is not able to produce a valid signature $\bar{\sigma}_{i,1}$. Next, suppose the adversary $\mathcal{A}$ tries to impersonate $\mathbf{S}_i$ in Phase 2, then in Phase 2 Round 1, $\mathcal{A}$ generates $y_i = g^{K_i} \bmod p$. Next $\mathcal{A}$ has to generate the valid signature $\sigma_i = \mathcal{S}(\mathbf{S}_i||y_i)$. The validity of σ_i is checked by $\mathbf{C}$ in Phase 2 Round 2. If signature σ_i is not correct, then $\mathbf{C}$ rejects $\mathbf{S}_i$. As $\mathcal{A}$ does not know the RSA private key d_i of $\mathbf{S}_i$ and the RSA signature scheme is assumed to be secure, therefore $\mathcal{A}$ is not able to produce a valid signature σ_i . Therefore, $\mathcal{A}$ is not able to impersonate $\mathbf{S}_i$ to establish the subgroup key K_i and final group key K .

Next, we consider $\mathcal{A}$ tries to impersonate user $\mathbf{U}_{i,j}$. Then in Phase 1 Round 2, $\mathcal{A}$ generates $z_{i,j} = g^{r_{i,j}} \bmod p$. Next, $\mathcal{A}$ has to compute the valid signature $\sigma_{i,j} = \mathcal{S}(\mathbf{U}_{i,j}||z_{i,j}||\tau_i||0)$. The validity of $\sigma_{i,j}$ is checked by $\mathbf{S}_i$ in Phase 1 Round 3. If the signature $\sigma_{i,j}$ is not correct, then $\mathbf{S}_i$ rejects $\mathbf{U}_{i,j}$. As $\mathcal{A}$ does not know the RSA private key $d_{i,j}$ of $\mathbf{U}_{i,j}$ and the RSA signature scheme is assumed to be secure, therefore $\mathcal{A}$ is not able to produce a valid signature $\sigma_{i,j}$. Next suppose $\mathcal{A}$ tries to impersonate $\mathbf{U}_{i,j}$ in Phase 1 Round 4, then $\mathcal{A}$ generates $\kappa_{i,j} \in \{0,1\}^\lambda$ and $X_{i,j} = L_{i,j} \oplus R_{i,j}$ where $L_{i,j} = \mathbf{H}(z_{i,j-1}^{r_{i,j}}||\mathbf{SID}_i)$, $R_{i,j} = \mathbf{H}(z_{i,j+1}^{r_{i,j}}||\mathbf{SID}_i)$ and $\mathbf{SID}_i = \mathbf{H}(\tau_i||\mathbf{U}_{i,1}||\mathbf{U}_{i,2}||\dots||\mathbf{U}_{i,n_i})$. Then $\mathcal{A}$

has to compute the valid signature $\bar{\sigma}_{i,j} = \mathcal{S}(\mathbf{U}_{i,j} || \kappa_{i,j} || X_{i,j} || \tau_i || 0)$. The validity of $\bar{\sigma}_{i,j}$ is checked by $\mathbf{S}_i$ in Phase 1 Round 5. If the signature $\bar{\sigma}_{i,j}$ is not correct, then $\mathbf{S}_i$ rejects $\mathbf{U}_{i,j}$. As $\mathcal{A}$ does not know the RSA private key $d_{i,j}$ of $\mathbf{U}_{i,j}$ and the RSA signature scheme is assumed to be secure, therefore $\mathcal{A}$ is not able to produce a valid signature $\bar{\sigma}_{i,j}$. Therefore, $\mathcal{A}$ is not able to impersonate $\mathbf{U}_{i,j}$ to establish the subgroup key K_i .

Finally, we consider $\mathcal{A}$ tries to impersonate main controller $\mathbf{C}$. Then in Phase 2 Round 2, $\mathcal{A}$ generates $T_i = y_i^{r_c} \bmod p$, final group key $K = (y_c \cdot \prod_{i=1}^{n_c} y_i)^{r_c} \bmod p$, $\theta_c = \mathbf{H}(K || y_c)$ and $\mathbf{c}_i = \mathcal{E}_{T_i}(K)$ for $1 \leq i \leq n_c$. Then $\mathcal{A}$ has to compute the valid signature $\sigma_c = \mathcal{S}(\mathbf{C} || y_c || \theta_c)$. The validity of σ_c is checked by $\mathbf{S}_i$ for $1 \leq i \leq n_c$ in Phase 3 Round 1. If signature σ_c is not correct, then $\mathbf{S}_i$ rejects $\mathbf{C}$. As $\mathcal{A}$ does not know the RSA private key d_c of $\mathbf{C}$ and the RSA signature scheme is assumed to be secure, therefore $\mathcal{A}$ is not able to produce a valid signature σ_c . Therefore, $\mathcal{A}$ is not able to impersonate $\mathbf{C}$ to establish the final group key K .

As $\mathcal{A}$ is not able to impersonate main controller $\mathbf{C}$ or subgroup controller $\mathbf{S}_i = \mathbf{U}_{i,1}$ ($1 \leq i \leq n_c$) or user $\mathbf{U}_{i,j}$ ($1 \leq i \leq n_c$ and $2 \leq j \leq n_i$) to establish subgroup keys K_i ($1 \leq i \leq n_c$) and final group key K , hence we shall now consider case (b) as follows: Suppose the adversary $\mathcal{A}$ tries to obtain the subgroup keys K_i ($1 \leq i \leq n_c$) from the up flow information sent. Then Phase 1 (without the signature information sent) of the Initial GKA protocol is just the VK-GKA protocol. By Theorem 27, it is shown that the VK-GKA protocol is secure, therefore, the adversary $\mathcal{A}$ is not able to obtain the subgroup keys K_i for $1 \leq i \leq n_c$ from the up flow information. Next, suppose the adversary $\mathcal{A}$ tries to obtain the final group key K from the up flow information sent. Then Phase 2 (without the signature information sent) of the Initial GKA protocol is just the VSTAR protocol where Phase 2 Round 1, Phase 2 Round 2 and Phase 3 Round 1 is the same as Round 1, Round 2 and Computation of Key of the VSTAR protocol respectively. By Theorem 28, it is shown that the VSTAR protocol is secure, hence the adversary $\mathcal{A}$ is not able to obtain the final group key K from the up flow information. ■

Next, we examine the down flow information (Phase 3) of the Initial GKA protocol as follows.

Theorem 30 *Given the down flow information (Phase 3), an adversary $\mathcal{A}$ cannot obtain the final group key K in the Initial GKA protocol provided the RSA signature scheme and symmetric key encryption scheme SKE are secure.*

Proof: First, we consider if the adversary $\mathcal{A}$ is able to impersonate S_i ($1 \leq i \leq n_c$) in Phase 3. Then in Phase 3 Round 1, $\mathcal{A}$ generates $\bar{c}_i = \mathcal{E}_{K_i}(K||S_i)$ and $\theta_i = H(K_i||K||\tau_i)$. Next, $\mathcal{A}$ needs to compute valid signature $\bar{\sigma}_i = \mathcal{S}(S_i||\tau_i||\theta_i||0)$. The validity of $\bar{\sigma}_i$ is checked by $U_{i,j}$ for $2 \leq j \leq n_i$ in Phase 3 Round 2. If the signature $\bar{\sigma}_i$ is not correct, then $U_{i,j}$ rejects S_i . As $\mathcal{A}$ does not know the RSA private key d_i of S_i and the RSA signature scheme is assumed to be secure, therefore $\mathcal{A}$ is not able to produce a valid signature $\bar{\sigma}_i$. Therefore, $\mathcal{A}$ is not able to impersonate S_i in Phase 3. Next we consider if the adversary $\mathcal{A}$ is able to obtain the final group key K from the down flow information, that is, $\bar{c}_i = \mathcal{E}_{K_i}(K||S_i)$ for $1 \leq i \leq n_c$ sent in Phase 3. The final group key K is the plaintext of $\bar{c}_i = \mathcal{E}_{K_i}(K||S_i)$ for $1 \leq i \leq n_c$. As $\mathcal{A}$ does not know the subgroup key K_i and the symmetric key encryption scheme SKE is assumed to be secure, therefore it is not possible for the adversary $\mathcal{A}$ to obtain the final group key K from $\bar{c}_i$. Hence, $\mathcal{A}$ is not able to obtain final group key K from the down flow information (Phase 3). ■

Theorem 31 *Assume that the computation of RSA signature verification is efficient, then the proposed Initial GKA protocol is resistant to DoS attack and insider attack.*

Proof: Suppose an adversary launches a DoS attack on the proposed Initial GKA protocol which stops the GKA from producing the final group key. First, the subgroup controller S_i in the proposed protocol is able to detect the invalid signatures or certificates in Phase 1 Round 2 by verifying the signatures and certificates. As both signature and certificate verifications are using RSA signature scheme, hence the verification is very fast (0.22ms, even for 1000 users, it only take 0.22s to verify all the users). S_i will inform his members of the invalid users and the protocol still continues. If the attacker tries to attack on Phase 1 Round 4, the proposed protocol is also able to detect the invalid signature and eliminate the invalid users. Then the protocol will still continue to Phase 2. The main controller C is able to detect the invalid signature and certificate in Phase 2 Round 2 by verifying the signature and certificates. C will reject the invalid subgroup controllers and the protocol can still proceed until the final group key K is formed. Hence, the attacker will not be able to stop the proposed Initial GKA protocol from forming the final group key K . Furthermore, upon receiving both the signature and certificate in the Initial GKA protocol, C , S_i and $U_{i,j}$ will first use the public key in the certificate to verify the signature. If signature is not valid, then there is no need to verify the certificate. Hence, even if the DoS attacker sends a valid certificate but

invalid signature, $\mathbf{C}$, $\mathbf{S}_i$ and $\mathbf{U}_{i,j}$ do not have to spend extra time to verify the certificate which is valid as the invalid signature is verified first. Next, suppose malicious group members try to launch insider attacks to disrupt and halt the Initial GKA Protocol by sending wrong information that causes an invalid XOR equation in Phase 1 Round 5, the detection protocol described in Section 9.3 is able to detect and block these malicious group members. The proposed protocol will then continue to establish the subgroup key K_i and the final group key K , thus preventing the insider attack. Hence, the Initial GKA protocol is able to resist DoS attack and insider attack. ■

9.4.2 Security Analysis of Join Protocol

We first show that the Join Protocol achieves backward secrecy, that is, it is not possible for new users $\mathbf{U}_{J,k}$ for $n_J + 1 \leq k \leq n_J + m$ to obtain the old subgroup key K_J and old final group key K from the Join Protocol. Next, we show that the adversary who eavesdrops on the communication is unable to obtain the new subgroup key K'_J and new final group key K' from the available public information. Finally we show that the Join Protocol is resistant to DoS attacks and insider attacks.

Theorem 32 *The Join protocol achieves backward secrecy provided the CDH problem is difficult.*

Proof: Suppose the new user $\mathbf{U}_{J,k}$ for $n_J + 1 \leq k \leq n_J + m$ tries to obtain the previous subgroup key $K_J = \mathbf{H}(\kappa_{J,1} || \kappa_{J,2} || \cdots || \kappa_{J,n_J} || 0)$ from the Join Protocol, then $\mathbf{U}_{J,k}$ first obtains $K'_J = \mathbf{H}(\kappa'_{J,1} || \kappa'_{J,n_J+1} || \cdots || \kappa'_{J,n_J+m} || v)$ from the Join Protocol. As $\kappa'_{J,1}$ and $\kappa_{J,1}$ are random and independent to each other, therefore K'_J and K_J are independent to each other as the output of the hash function $\mathbf{H}$ is independent. Since $\mathbf{U}_{J,k}$ knows $\kappa'_{J,1}$, $\kappa'_{J,k}$ for $n_J + 1 \leq k \leq n_J + m$ and also $\kappa_{J,j}$ for $2 \leq j \leq n_J$ from the previous GKA, we consider whether $\mathbf{U}_{J,k}$ can obtain $\kappa_{J,1}$ from the information he knows in the Join Protocol and compute the previous subgroup key K_J . As τ'_J and τ_J are random and independent to each other, therefore, $\mathbf{SID}'_J = \mathbf{H}(\tau'_J || \mathbf{U}_{J,1} || \cdots || \mathbf{U}_{J,n_J+m})$ and $\mathbf{SID}_J = \mathbf{H}(\tau_J || \mathbf{U}_{J,1} || \mathbf{U}_{J,2} || \cdots || \mathbf{U}_{J,n_J})$ are independent to each other as the output of the hash function $\mathbf{H}$ is independent. Furthermore, as $r'_{J,1}$ and $r_{J,1}$ are random and independent to each other, therefore $g^{r'_{J,1}r'_{J,n_J+1}}$ and $g^{r_{J,1}r_{J,2}}$ are independent to each other. Hence, $R'_{J,1} = \mathbf{H}(g^{r'_{J,1}r'_{J,n_J+1}} || \mathbf{SID}'_J)$ and $R_{J,1} = \mathbf{H}(g^{r_{J,1}r_{J,2}} || \mathbf{SID}_J)$ are independent to each other as the output of the hash function $\mathbf{H}$ is independent. Hence, $\widehat{X}'_J = \kappa'_{J,1} \oplus R'_{J,1}$ and $\widehat{X}_J = \kappa_{J,1} \oplus R_{J,1}$ are independent to each other and $\mathbf{U}_{J,k}$ is not able to obtain $\kappa_{J,1}$ from $\widehat{X}_J$ using the information $\widehat{X}'_J$, $\kappa'_{J,1}$ and $R'_{J,1}$ that he

knows. Furthermore, if $U_{J,k}$ can obtain $g^{r_{J,1}r_{J,2}}$ from $z_{J,1} = g^{r_{J,1}}$ and $z_{J,2} = g^{r_{J,2}}$, then $U_{J,k}$ can obtain $R_{J,1} = H(g^{r_{J,1}r_{J,2}}||\text{SID}_J)$ and $\kappa_{J,1} = \widehat{X}_J \oplus R_{J,1}$ and compute the previous subgroup key $K_J = H(\kappa_{J,1}||\kappa_{J,2}||\dots||\kappa_{J,n_J}||0)$. But it is not possible for $U_{J,k}$ to obtain $g^{r_{J,1}r_{J,2}}$ from $z_{J,1}$ and $z_{J,2}$ as this is the CDH problem and is assumed to be difficult.

Next, we consider new user $U_{J,k}$ for $n_J + 1 \leq k \leq n_J + m$ tries to obtain previous group key $K = g^{r_{\text{c}}r_{\text{c}}+r_{\text{c}}K_1+r_{\text{c}}K_2+\dots+r_{\text{c}}K_J+\dots+r_{\text{c}}K_{n_{\text{c}}}}$. Then $U_{J,k}$ first obtains new final group key $K' = g^{r'_{\text{c}}r'_{\text{c}}+r'_{\text{c}}K_1+r'_{\text{c}}K_2+\dots+r'_{\text{c}}K'_J+\dots+r'_{\text{c}}K_{n_{\text{c}}}}$. As r'_{c} and r_{c} are random and independent to each other and K'_J and K_J are independent to each other, therefore $g^{r'_{\text{c}}r'_{\text{c}}}$ and $g^{r_{\text{c}}r_{\text{c}}}$, $g^{r'_{\text{c}}K_i}$ and $g^{r_{\text{c}}K_i}$ for $1 \leq i \neq J \leq n_{\text{c}}$, and $g^{r'_{\text{c}}K'_J}$ and $g^{r_{\text{c}}K_J}$ are independent to each other respectively. Hence, K' and K are independent to each other. Furthermore, if $U_{J,k}$ can obtain $g^{r_{\text{c}}r_{\text{c}}}$ from $y_{\text{c}} = g^{r_{\text{c}}}$ and $g^{r_{\text{c}}K_i}$ from $y_{\text{c}} = g^{r_{\text{c}}}$ and $y_i = g^{K_i}$ for $1 \leq i \leq n_{\text{c}}$ respectively, then $U_{J,k}$ can obtain the previous group key $K = g^{r_{\text{c}}r_{\text{c}}+r_{\text{c}}K_1+r_{\text{c}}K_2+\dots+r_{\text{c}}K_J+\dots+r_{\text{c}}K_{n_{\text{c}}}}$. But this is not possible for $U_{J,k}$ to obtain $g^{r_{\text{c}}r_{\text{c}}}$ from y_{c} and $g^{r_{\text{c}}K_i}$ from y_{c} and y_i respectively as this is the CDH problem and is assumed to be difficult. Hence, the Join Protocol achieves backward secrecy. ■

The Join protocol also has to ensure that the adversary who eavesdrops on the communication cannot obtain new subgroup key K'_J and new final group key K' .

Theorem 33 *An adversary $\mathcal{A}$ is not able to obtain the new subgroup key K'_J and new final group key K' in the Join Protocol provided the VK-GKA protocol, VSTAR protocol, the RSA signature scheme and the symmetric key encryption scheme SKE are secure.*

Proof: In this proof, we consider two cases, that is (a) the adversary $\mathcal{A}$ tries to impersonate subgroup controller S_J , subgroup controller S_i for $1 \leq i \neq J \leq n_{\text{c}}$, new users $U_{J,k}$ for $n_J + 1 \leq k \leq n_J + m$, user U_{J,n_J} or main controller C to establish the new subgroup key K'_J and new final group key K' (we do not consider $\mathcal{A}$ tries to impersonate users $U_{i,j}$ for $1 \leq i \neq J \leq n_{\text{c}}$ and $2 \leq j \leq n_i$ and $U_{J,j}$ for $2 \leq j \leq n_J - 1$ as $U_{i,j}$ and $U_{J,j}$ does not send any information in the Join Protocol) and (b) the adversary $\mathcal{A}$ tries to obtain the new subgroup key K'_J and new final group key K' from the public information in the Join Protocol. For case (a), we first consider $\mathcal{A}$ tries to impersonate subgroup controller S_J . Then in Phase 1 Round 1, $\mathcal{A}$ generates $\tau'_J = H(\kappa'_{J,1}||v)$. Next in Phase 1 Round 3, $\mathcal{A}$ first generates $z'_{J,1} = g^{r'_{J,1}} \bmod p$, then $\mathcal{A}$ needs to compute the valid signature $\sigma'_{J,1} = \mathcal{S}(S_J||\tau'_J||z'_{J,1}||\text{BL}_Z||v)$. The validity of signature $\sigma'_{J,1}$ is checked by $U_{J,k}$ for

$n_J + 1 \leq k \leq n_J + m$ and U_{J,n_J} in Phase 1 Round 4. If the signature $\sigma'_{J,1}$ is not correct, then $U_{J,k}$ and U_{J,n_J} rejects S_J . As $\mathcal{A}$ does not know the RSA private key d_J of S_J and the RSA signature scheme is assumed to be secure, therefore $\mathcal{A}$ is not able to produce a valid signature $\sigma'_{J,1}$. Next suppose $\mathcal{A}$ tries to impersonate S_J in Phase 1 Round 5, then $\mathcal{A}$ first generates $X'_{J,1} = L'_{J,1} \oplus R'_{J,1}$ and $\widehat{X}'_J = \kappa'_{J,1} \oplus R'_{J,1}$ where $L'_{J,1} = H((z'_{J,n_J})^{r'_{J,1}} || \text{SID}'_J)$ and $R'_{J,1} = H((z'_{J,n_J+1})^{r'_{J,1}} || \text{SID}'_J)$. Then, $\mathcal{A}$ has to compute the valid signature $\bar{\sigma}'_{J,1} = \mathcal{S}(S_J || \tau'_J || \widehat{X}'_J || X'_{J,1} || v)$. The validity of signature $\bar{\sigma}'_{J,1}$ is checked by $U_{J,k}$ for $n_J + 1 \leq k \leq n_J + m$ and U_{J,n_J} in Phase 1 Round 6. If the signature $\bar{\sigma}'_{J,1}$ is not correct, then $U_{J,k}$ and U_{J,n_J} rejects S_J . As $\mathcal{A}$ does not know the RSA private key d_J of S_J and the RSA signature scheme is assumed to be secure, therefore $\mathcal{A}$ is not able to produce a valid signature $\bar{\sigma}'_{J,1}$. Next, suppose the adversary $\mathcal{A}$ tries to impersonate S_J in Phase 2. Then in Phase 2 Round 1, $\mathcal{A}$ generates $y'_J = g^{K'_J} \bmod p$. Next $\mathcal{A}$ has to generate the valid signature $\sigma'_J = \mathcal{S}(S_J || y'_J)$. The validity of σ'_J is checked by C in Phase 2 Round 2. If signature σ'_J is not correct, then C rejects S_J . As $\mathcal{A}$ does not know the RSA private key d_J of S_J and the RSA signature scheme is assumed to be secure, therefore $\mathcal{A}$ is not able to produce a valid signature σ'_J . Next, we consider $\mathcal{A}$ tries to impersonate S_J in Phase 3. Then in Phase 3 Round 1, $\mathcal{A}$ generates $\bar{c}'_J = \mathcal{E}_{K'_J}(K' || S_J)$ and $\bar{c}''_J = \mathcal{E}_{K_J}(K'_J || K' || S_J)$ and $\theta'_J = H(K'_J || K' || \tau'_J)$. Next, $\mathcal{A}$ needs to compute valid signature $\bar{\sigma}'_J = \mathcal{S}(S_J || \tau'_J || \theta'_J || v)$. The validity of $\bar{\sigma}'_J$ is checked by $U_{J,j}$ for $2 \leq j \leq n_J$ and $U_{J,k}$ for $n_J + 1 \leq k \leq n_J + m$ in Phase 3 Round 2. If the signature $\bar{\sigma}'_J$ is not correct, then $U_{J,j}$ and $U_{J,k}$ rejects S_J . As $\mathcal{A}$ does not know the RSA private key d_J of S_J and the RSA signature scheme is assumed to be secure, therefore $\mathcal{A}$ is not able to produce a valid signature $\bar{\sigma}'_J$. Therefore, $\mathcal{A}$ is not able to impersonate S_J to establish the subgroup key K'_J and final group key K' .

Next, we consider $\mathcal{A}$ wishes to impersonate subgroup controller S_i for $1 \leq i \neq J \leq n_c$. As S_i for $1 \leq i \neq J \leq n_c$ does not send any information in Phase 1 and 2, hence we only consider $\mathcal{A}$ tries to impersonate S_i in Phase 3. Then in Phase 3 Round 1, $\mathcal{A}$ generates $\bar{c}'_i = \mathcal{E}_{K'_i}(K' || S_i)$ and $\theta'_i = H(K'_i || K' || \tau_i)$. Next, $\mathcal{A}$ needs to compute valid signature $\bar{\sigma}'_i = \mathcal{S}(S_i || \tau_i || \theta'_i || v)$. The validity of $\bar{\sigma}'_i$ is checked by $U_{i,j}$ for $2 \leq j \leq n_i$ in Phase 3 Round 2. If the signature $\bar{\sigma}'_i$ is not correct, then $U_{i,j}$ rejects S_i . As $\mathcal{A}$ does not know the RSA private key d_i of S_i and the RSA signature scheme is assumed to be secure, therefore $\mathcal{A}$ is not able to produce a valid signature $\bar{\sigma}'_i$. Therefore, $\mathcal{A}$ is not able to impersonate S_i .

Next, consider if $\mathcal{A}$ tries to impersonate new user $U_{J,k}$ for $n_J + 1 \leq k \leq n_J + m$ or existing user U_{J,n_J} . Then in Phase 1 Round 2, $\mathcal{A}$ generates $z'_{J,k} = g^{r'_{J,k}} \bmod p$ ($z'_{J,n_J} = g^{r'_{J,n_J}} \bmod p$). Next, $\mathcal{A}$ has to compute the valid signature $\sigma'_{J,k} = \mathcal{S}(U_{J,k} || z'_{J,k} || \tau'_J || v)$ ($\sigma'_{J,n_J} = \mathcal{S}(U_{J,n_J} || z'_{J,n_J} || \tau'_J || v)$). The validity of $\sigma'_{J,k}$ (σ'_{J,n_J}) is checked by S_J in Phase 1 Round 3. If the signature $\sigma'_{J,k}$ (σ'_{J,n_J}) is not correct, then S_J rejects $U_{J,k}$ (U_{J,n_J}). As $\mathcal{A}$ does not know the RSA private key $d_{J,k}$ (d_{J,n_J}) of $U_{J,k}$ (U_{J,n_J}) and the RSA signature scheme is assumed to be secure, therefore $\mathcal{A}$ is not able to produce a valid signature $\sigma'_{J,k}$ (σ'_{J,n_J}). Next suppose $\mathcal{A}$ tries to impersonate $U_{J,k}$ (U_{J,n_J}) in Phase 1 Round 4, then $\mathcal{A}$ generates $\kappa'_{J,k} \in \{0, 1\}^\lambda$ ($\kappa'_{J,n_J} \in \{0, 1\}^\lambda$) and $X'_{J,k} = L'_{J,k} \oplus R'_{J,k}$ ($X'_{J,n_J} = L'_{J,n_J} \oplus R'_{J,n_J}$), where $L'_{J,k} = H((z'_{J,k-1})^{r'_{J,k}} || \text{SID}'_J)$ ($L'_{J,n_J} = H((z'_{J,n_J+m})^{r'_{J,n_J}} || \text{SID}'_J)$), $R'_{J,k} = H((z'_{J,k+1})^{r'_{J,k}} || \text{SID}'_J)$ ($R'_{J,n_J} = H((z'_{J,1})^{r'_{J,n_J}} || \text{SID}'_J)$) and $\text{SID}'_J = H(\tau'_J || U_{J,1} || \dots || U_{J,n_J+m} || U_{J,n_J})$. Then $\mathcal{A}$ has to compute the valid signature $\bar{\sigma}'_{J,k} = \mathcal{S}(U_{J,k} || \kappa'_{J,k} || X'_{J,k} || \tau'_J || v)$ ($\bar{\sigma}'_{J,n_J} = \mathcal{S}(U_{J,n_J} || \kappa'_{J,n_J} || X'_{J,n_J} || \tau'_J || v)$). The validity of $\bar{\sigma}'_{J,k}$ ($\bar{\sigma}'_{J,n_J}$) is checked by S_J in Phase 1 Round 5. If the signature $\bar{\sigma}'_{J,k}$ ($\bar{\sigma}'_{J,n_J}$) is not correct, then S_J rejects $U_{J,k}$ (U_{J,n_J}). As $\mathcal{A}$ does not know the RSA private key $d_{J,k}$ (d_{J,n_J}) of $U_{J,k}$ (U_{J,n_J}) and the RSA signature scheme is assumed to be secure, therefore $\mathcal{A}$ is not able to produce a valid signature $\bar{\sigma}'_{J,k}$ ($\bar{\sigma}'_{J,n_J}$). Therefore, $\mathcal{A}$ is not able to impersonate $U_{J,k}$ (U_{J,n_J}).

Next, we consider $\mathcal{A}$ tries to impersonate C . Then in Phase 2 Round 1, $\mathcal{A}$ generates $T'_J = (y'_J)^{r'_c} \bmod p$ and $T'_i = y_i^{r'_c} \bmod p$ for $1 \leq i \neq J \leq n_c$, final group key $K' = (y'_c \cdot y'_J \cdot \prod_{i=1, i \neq J}^{n_c} y_i)^{r'_c}$, $\theta'_c = H(K' || y'_c)$, $c'_J = \mathcal{E}_{T'_J}(K')$ and $c'_i = \mathcal{E}_{T'_i}(K')$ for $1 \leq i \neq J \leq n_c$. Then $\mathcal{A}$ has to compute the valid signature $\sigma'_c = \mathcal{S}(C || y'_c || \theta'_c)$. The validity of σ'_c is checked by S_i (including S_J) for $1 \leq i \leq n_c$ in Phase 3 Round 1. If signature σ'_c is not correct, then S_i (including S_J) rejects C . As $\mathcal{A}$ does not know the RSA private key d_c of C and the RSA signature scheme is assumed to be secure, therefore $\mathcal{A}$ is not able to produce a valid signature σ'_c . Therefore, $\mathcal{A}$ is not able to impersonate C to establish the final group key K . Hence it is not possible for $\mathcal{A}$ to obtain new subgroup key K'_J and new final group key K' in case (a).

Next we consider case (b) as follows: as the adversary $\mathcal{A}$ is not a joining user, therefore, $\mathcal{A}$ can only obtain the public exchange information in the Join Protocol. First we consider if $\mathcal{A}$ can obtain the new subgroup key K'_J from the public information in Phase 1. But Phase 1 of the Join Protocol (without the signature information sent) is just the VK-GKA protocol (described in Section 9.4.1) executed by subgroup controller S_J , new users $U_{J,k}$ for $n_J + 1 \leq k \leq n_J + m$ and existing user U_{J,n_J} .

to establish the new subgroup key K'_J . By Theorem 27, it is shown that the VK-GKA protocol is secure. Therefore, the adversary $\mathcal{A}$ is not able to obtain the subgroup key K'_J from Phase 1 of the Join Protocol. Next, suppose the adversary $\mathcal{A}$ tries to obtain the final group key K' from Phase 2 of the Join Protocol. But Phase 2 of the Join Protocol (without the signature information sent) is just the VSTAR protocol (described in Section 9.4.1) where subgroup controller $\mathbf{S}_J$ computes and sends $y'_J = g^{K'_J}$ in Phase 2 Round 1 while $\mathbf{S}_i$ for $1 \leq i \neq J \leq n_c$ uses $y_i = g^{K_i}$ that $\mathbf{C}$ knows from previous GKA, and Phase 2 Round 2 and Phase 3 Round 1 are the same as Round 2 and Computation of Key of VSTAR protocol respectively. By Theorem 28, it is shown that the VSTAR protocol is secure. Hence the adversary $\mathcal{A}$ is not able to obtain the final group key K' from Phase 2 of the Join Protocol. Next, we consider the adversary $\mathcal{A}$ tries to obtain the new subgroup key K'_J and new final group key K' from the public exchange information, that is, $\bar{c}'_J$, $\bar{c}''_J$ and $\bar{c}'_i$ for $1 \leq i \neq J \leq n_c$, in Phase 3 of the Join Protocol. The new subgroup key K'_J and new final group key K' are the plaintext of $\bar{c}''_J = \mathcal{E}_{K_J}(K'_J || K' || \mathbf{S}_J)$ and the new group key K' is the plaintext of $\bar{c}'_J = \mathcal{E}_{K'_J}(K' || \mathbf{S}_J)$ and $\bar{c}'_i = \mathcal{E}_{K_i}(K' || \mathbf{S}_i)$ for $1 \leq i \neq J \leq n_c$. As $\mathcal{A}$ does not know the new subgroup key K'_J , existing subgroup keys K_J and K_i for $1 \leq i \neq J \leq n_c$, and the symmetric key encryption scheme SKE is assumed to be secure, therefore $\mathcal{A}$ is not able to obtain the new subgroup key K'_J and new final group key K' from $\bar{c}''_J$ and the new group key K' from $\bar{c}'_J$ and $\bar{c}'_i$ for $1 \leq i \neq J \leq n_c$ respectively. Hence, $\mathcal{A}$ is not able to obtain the new subgroup key K'_J and new final group key K' in the Join Protocol. ■

Theorem 34 *Assume that the computation of RSA signature verification is efficient, then the proposed Join protocol is resistant to DoS attack and insider attack.*

Proof: Suppose an adversary launches a DoS attack on the proposed Join protocol which stops the GKA from producing the new final group key. In Phase 1 of the Join Protocol, we only consider the DoS attacker launch DoS attack on subgroup controller $\mathbf{S}_J$ as the other subgroup controllers $\mathbf{S}_i$ for $1 \leq i \neq J \leq n_c$ do not send any information in Phase 1. Then in Phase 1 Round 2, $\mathbf{S}_J$ is able to detect the invalid signatures or certificates by verifying the signatures and certificates. As mentioned in Theorem 31, both signature and certificate verifications are using RSA signature scheme. Hence the verification is very fast and $\mathbf{S}_J$ will inform $\mathbf{U}_{J,k}$ for $n_J + 1 \leq k \leq n_J + m$ and $\mathbf{U}_{J,n_J}$ of the invalid users and the Join protocol still continues. If the attacker tries to attack on

Phase 1 Round 4, then S_J is also able to detect the invalid signature and eliminate the invalid users. Then the protocol will still continue to Phase 2. Furthermore, upon receiving both the signature and certificate in the Join protocol, S_J and new users $U_{J,k}$ for $n_J + 1 \leq k \leq n_J + m$ will first use the public key in the certificate to verify the signature. If signature is not valid, then there is no need to verify the certificate. Hence, even if the DoS attacker sends a valid certificate but invalid signature, S_J and new users $U_{J,k}$ do not have to spend extra time to verify the certificate which is valid as the invalid signature is verified first. In Phase 2 of the Join Protocol, only subgroup controller S_J sends information to main controller C in Phase 2 Round 1 to establish the new final group key while the other subgroup controllers S_i for $1 \leq i \neq J \leq n_c$ does not send any information in Phase 2. Hence if the DoS attacker tries to attack in Phase 2 Round 1, then in Phase 2 Round 2, the main controller C is able to detect and reject all invalid messages and invalid signatures sent by the DoS attacker and only accept the valid message and valid signature sent by subgroup controller S_J . Then the Join Protocol will proceed until the new final group key is formed. Hence, the attacker will not be able to stop the proposed Join protocol from forming the new final group key K' . Next, suppose malicious group members try to launch insider attacks to disrupt and halt the Join Protocol by sending wrong information that causes an invalid XOR equation in subgroup $\mathcal{G}'_J$ in Phase 1 Round 5, the detection protocol described in Section 9.3 is able to detect and block these malicious group members in subgroup $\mathcal{G}'_J$. The proposed protocol will then continue to establish the new subgroup key K'_J and the new final group key K' , thus preventing the insider attack. The malicious group members are not able to launch insider attacks in subgroups $\mathcal{G}_i \neq \mathcal{G}'_J$ for $1 \leq i \neq J \leq n_c$ in the Join Protocol as subgroups $\mathcal{G}_i \neq \mathcal{G}'_J$ do not send and receive any information in Phase 1 of the Join Protocol. Hence, the Join protocol is able to resist DoS attack and insider attack. ■

9.4.3 Security Analysis of Leave Protocol

In this section, we first show that the Leave protocol achieves forward secrecy, that is, leaving user $U_{L,k}$ for $1 \leq k \leq \ell$ is not able to obtain the new subgroup key K'_L and new final group key K' from the Leave Protocol. Next we show that an adversary is not able to obtain the new subgroup key K'_L and new final group key K' from the Leave Protocol. Finally we show that the Leave Protocol is resistant to DoS attacks and insider attacks.

Theorem 35 *The Leave Protocol achieves forward secrecy provided that the VK-GKA protocol, VSTAR protocol and the symmetric key encryption scheme SKE are secure and the CDH problem is difficult.*

Proof: First, we consider a leaving user U_{L,l_k} ($1 \leq k \leq \ell$) tries to obtain the new subgroup key $K'_L = H(\kappa'_{L,1} || \kappa'_{L,2} || \cdots || \kappa'_{L,n_L-\ell} || v)$. Then leaving user U_{L,l_k} knows $\kappa_{L,j}$ for $1 \leq j \leq n_L$, previous subgroup key $K_L = H(\kappa_{L,1} || \kappa_{L,2} || \cdots || \kappa_{L,n_L} || 0)$, $R_{L,j} = H(z_{L,j+1}^{r_{L,j}} || \text{SID}_L)$ and $L_{L,j} = H(z_{L,j-1}^{r_{L,j}} || \text{SID}_L)$ for $1 \leq j \leq n_L$, where $\text{SID}_L = H(\tau_L || U_{L,1} || U_{L,2} || \cdots || U_{L,n_L})$. U_{L,l_k} also knows $\kappa'_{L,j}$ for $2 \leq j \leq n_L - \ell$ from the Leave Protocol. But the new subgroup key is $K'_L = H(\kappa'_{L,1} || \kappa'_{L,2} || \cdots || \kappa'_{L,n_L-\ell} || v)$. As $\kappa'_{L,1}$ is random and independent of $\kappa_{L,1}$ from the previous GKA, hence K'_L and K_L are independent to each other as the output of the hash function H is independent. As each remaining user $V_{L,k} \in \mathcal{N}(\mathcal{R})$, who is a left or right neighbor of leaving users, selects new random $r'_{L,k} \in Z_q^*$ and computes $z'_{L,k} = g^{r'_{L,k}}$ in Phase 1 Round 2 of the Leave Protocol, therefore, $r'_{L,k}$ and $r_{L,k}$ are random and independent to each other. Hence, $g^{r'_{L,k}}$ and $g^{r_{L,k}}$, $g^{r'_{L,k}r'_{L,k+1}}$ and $g^{r_{L,k}r_{L,k+1}}$, and $g^{r'_{L,k}r'_{L,k-1}}$ and $g^{r_{L,k}r_{L,k-1}}$ (where $r'_{L,k+1} = r_{L,k+1}$ if $V_{L,k+1} \notin \mathcal{N}(\mathcal{R})$ and $r'_{L,k-1} = r_{L,k-1}$ if $V_{L,k-1} \notin \mathcal{N}(\mathcal{R})$) are independent to each other. Furthermore, as τ'_L and τ_L are random and independent to each other, hence $\text{SID}'_L = H(\tau'_L || V_{L,1} || V_{L,2} || \cdots || V_{L,n_L-\ell})$ and $\text{SID}_L = H(\tau_L || U_{L,1} || U_{L,2} || \cdots || U_{L,n_L})$ are independent to each other as the output of the hash function H is independent. Therefore, $R'_{L,j} = H(g^{r'_{L,j}r'_{L,j+1}} || \text{SID}'_L)$ and $R_{L,j} = H(g^{r_{L,j}r_{L,j+1}} || \text{SID}_L)$, and $L'_{L,j} = H(g^{r'_{L,j}r'_{L,j-1}} || \text{SID}'_L)$ and $L_{L,j} = H(g^{r_{L,j}r_{L,j-1}} || \text{SID}_L)$ (where $r'_{L,j} = r_{L,j}$ if $V_{L,j} \notin \mathcal{N}(\mathcal{R})$) are independent to each other respectively as the output of the hash function H is independent. Hence leaving user U_{L,l_k} is not able to obtain $R'_{L,j}$ and $L'_{L,j}$ for $1 \leq j \leq n_L - \ell$ and $\kappa'_{L,1}$ from $R_{L,j}$ and $L_{L,j}$ for $1 \leq j \leq n_L$ that he knows. On the other hand, if the leaving user U_{L,l_k} is able to obtain $g^{r'_{L,j}r'_{L,j+1}}$ from $z'_{L,j} = g^{r'_{L,j}}$ and $z'_{L,j+1} = g^{r'_{L,j+1}}$ for $1 \leq j \leq n_L - \ell$ (where $r'_{L,j} = r_{L,j}$ and $z'_{L,j} = z_{L,j}$ if $V_{L,j} \notin \mathcal{N}(\mathcal{R})$), then U_{L,l_k} can obtain $R'_{L,j} = H(g^{r'_{L,j}r'_{L,j+1}} || \text{SID}'_L)$ for $1 \leq j \leq n_L - \ell$ and $\kappa'_{L,1} = \widehat{X}'_L \oplus R'_{L,1}$ and compute the new subgroup key $K'_L = H(\kappa'_{L,1} || \kappa'_{L,2} || \cdots || \kappa'_{L,n_L-\ell} || v)$. But it is not possible for leaving user U_{L,l_k} to obtain $g^{r'_{L,j}r'_{L,j+1}}$ from $z'_{L,j}$ and $z'_{L,j+1}$ for $1 \leq j \leq n_L - \ell$ as this is the CDH problem and is assumed to be difficult. Hence it is not possible for leaving user U_{L,l_k} for $1 \leq k \leq \ell$ to obtain the new subgroup key K'_L from the information he knows in the previous GKA. This leads to the case that leaving user U_{L,l_k} ($1 \leq k \leq \ell$) tries to obtain the new subgroup key K'_L from the public

information sent in Phase 1 of the Leave Protocol. But Phase 1 of the Leave Protocol (without the signature information sent) is just the VK-GKA protocol (described in Section 9.4.1) executed by subgroup controller S_L and remaining users $V_{L,j}$ for $2 \leq j \leq n_L - \ell$ to establish the new subgroup key K'_L . By Theorem 27, it is shown that the VK-GKA protocol is secure. Therefore, the leaving user U_{L,l_k} is not able to obtain the subgroup key K'_L from Phase 1 of the Leave Protocol.

Next, suppose leaving user U_{L,l_k} ($1 \leq k \leq \ell$) tries to obtain the new final group key $K' = g^{r'_c r'_c + r'_c K_1 + r'_c K_2 + \dots + r'_c K'_L + \dots + r'_c K_{n_c}}$ from the previous group key $K = g^{r_c r_c + r_c K_1 + r_c K_2 + \dots + r_c K_L + \dots + r_c K_{n_c}}$. As r'_c and r_c are random and independent to each other and K'_L and K_L are independent to each other, therefore $g^{r'_c r'_c}$ and $g^{r_c r_c}$, $g^{r'_c K_i}$ and $g^{r_c K_i}$ for $1 \leq i \neq L \leq n_c$, and $g^{r'_c K'_L}$ and $g^{r_c K_L}$ are independent to each other respectively. Hence, new final group key K' and previous group key K are independent to each other. Then leaving user U_{L,l_k} only has the public information from Phase 2 of the Leave Protocol. But Phase 2 of the Leave Protocol (without the signature information sent) is just the VSTAR protocol (described in Section 9.4.1) where subgroup controller S_L computes and sends $y'_L = g^{K'_L}$ in Phase 2 Round 1 while S_i for $1 \leq i \neq L \leq n_c$ uses $y_i = g^{K_i}$ that C knows from previous GKA, and Phase 2 Round 2 and Phase 3 Round 1 are the same as Round 2 and Computation of Key of VSTAR protocol respectively. By Theorem 28, it is shown that the VSTAR protocol is secure. Hence the leaving user U_{L,l_k} is not able to obtain the final group key K' from Phase 2 of the Leave Protocol. Next, we consider the leaving user U_{L,l_k} tries to obtain the final group key K' from the public downflow information, that is, $\bar{c}'_L$ and $\bar{c}'_i$ for $1 \leq i \neq L \leq n_c$, in Phase 3 of the Leave Protocol. The new group key K' is the plaintext of $\bar{c}'_L = \mathcal{E}_{K'_L}(K' || S_L)$ and $\bar{c}'_i = \mathcal{E}_{K_i}(K' || S_i)$ for $1 \leq i \neq L \leq n_c$. As the leaving user U_{L,l_k} does not know new subgroup key K'_L and existing subgroup keys K_i for $1 \leq i \neq L \leq n_c$, and the symmetric key encryption scheme SKE is assumed to be secure, therefore it is also not possible for U_{L,l_k} to obtain K' from $\bar{c}'_L$ and $\bar{c}'_i$ for $1 \leq i \neq L \leq n_c$. Hence, the Leave Protocol provides backward secrecy. ■

Theorem 36 *An adversary $\mathcal{A}$ (not a leaving user U_{L,l_k} ($1 \leq k \leq \ell$)) cannot obtain the new subgroup key K'_L and new final group key K' in the Leave Protocol provided the VK-GKA protocol, the VSTAR protocol, the RSA signature scheme and the symmetric key encryption scheme SKE are secure.*

Proof: In this proof, we consider two cases, that is (a) the adversary $\mathcal{A}$ tries to impersonate subgroup controller $\mathbf{S}_L = \mathbf{V}_{L,1}$, subgroup controller $\mathbf{S}_i$ for $1 \leq i \neq L \leq n_c$, remaining users $\mathbf{V}_{L,j} \in \mathcal{G}'_L$ for $2 \leq j \leq n_L - \ell$ or main controller $\mathbf{C}$ to establish the new subgroup key K'_L and new final group key K' (we do not consider $\mathcal{A}$ tries to impersonate users $\mathbf{U}_{i,j}$ for $1 \leq i \neq L \leq n_c$ and $2 \leq j \leq n_i$ as $\mathbf{U}_{i,j}$ does not send any information in the Leave Protocol) and (b) $\mathcal{A}$ tries to obtain the new subgroup key K'_L and new final group key K' from the public information in the Leave Protocol.

For case (a), we first consider $\mathcal{A}$ tries to impersonate subgroup controller $\mathbf{S}_L$. Then in Phase 1 Round 1, $\mathcal{A}$ generates $\tau'_L = \mathbf{H}(\kappa'_{L,1}||v)$. Next in Phase 1 Round 3, $\mathcal{A}$ first generates $z'_{L,1} = g^{r'_{L,1}} \bmod p$, then $\mathcal{A}$ needs to compute the valid signature $\sigma'_{L,1} = \mathcal{S}(\mathbf{S}_L||\tau'_L||z'_{L,1}||\mathbf{BL}_Z||v)$. The validity of signature $\sigma'_{L,1}$ is checked by $\mathbf{V}_{L,j}$ for $2 \leq j \leq n_L - \ell$ in Phase 1 Round 4. If the signature $\sigma'_{L,1}$ is not correct, then $\mathbf{V}_{L,j}$ rejects $\mathbf{S}_L$. As $\mathcal{A}$ does not know the RSA private key d_L of $\mathbf{S}_L$ and the RSA signature scheme is assumed to be secure, therefore $\mathcal{A}$ is not able to produce a valid signature $\sigma'_{L,1}$. Next suppose $\mathcal{A}$ tries to impersonate $\mathbf{S}_L$ in Phase 1 Round 5, then $\mathcal{A}$ first generates $X'_{L,1} = L'_{L,1} \oplus R'_{L,1}$ and $\widehat{X}'_L = \kappa'_{L,1} \oplus R'_{L,1}$ where $L'_{L,1} = \mathbf{H}((z'_{L,n_L-\ell})^{r'_{L,1}}||\mathbf{SID}'_L)$ and $R'_{L,1} = \mathbf{H}((z'_{L,2})^{r'_{L,1}}||\mathbf{SID}'_L)$. Then, $\mathcal{A}$ has to compute the valid signature $\bar{\sigma}'_{L,1} = \mathcal{S}(\mathbf{S}_L||\tau'_L||\widehat{X}'_L||X'_{L,1}||v)$. The validity of signature $\bar{\sigma}'_{L,1}$ is checked by $\mathbf{V}_{L,j}$ for $2 \leq j \leq n_L - \ell$ in Phase 1 Round 6. If the signature $\bar{\sigma}'_{L,1}$ is not correct, then $\mathbf{V}_{L,j}$ rejects $\mathbf{S}_L$. As $\mathcal{A}$ does not know the RSA private key d_L of $\mathbf{S}_L$ and the RSA signature scheme is assumed to be secure, therefore $\mathcal{A}$ is not able to produce a valid signature $\bar{\sigma}'_{L,1}$. Next, suppose the adversary $\mathcal{A}$ tries to impersonate $\mathbf{S}_L$ in Phase 2. Then in Phase 2 Round 1, $\mathcal{A}$ generates $y'_L = g^{K'_L} \bmod p$. Next $\mathcal{A}$ has to generate the valid signature $\sigma'_L = \mathcal{S}(\mathbf{S}_L||y'_L)$. The validity of σ'_L is checked by $\mathbf{C}$ in Phase 2 Round 2. If signature σ'_L is not correct, then $\mathbf{C}$ rejects $\mathbf{S}_L$. As $\mathcal{A}$ does not know the RSA private key d_L of $\mathbf{S}_L$ and the RSA signature scheme is assumed to be secure, therefore $\mathcal{A}$ is not able to produce a valid signature σ'_L . Next, we consider $\mathcal{A}$ tries to impersonate $\mathbf{S}_L$ in Phase 3. Then in Phase 3 Round 1, $\mathcal{A}$ generates $\bar{c}'_L = \mathcal{E}_{K'_L}(K'||\mathbf{S}_L)$ and $\theta'_L = \mathbf{H}(K'_L||K'||\tau'_L)$. Next, $\mathcal{A}$ needs to compute valid signature $\bar{\sigma}'_L = \mathcal{S}(\mathbf{S}_L||\tau'_L||\theta'_L||v)$. The validity of $\bar{\sigma}'_L$ is checked by $\mathbf{V}_{L,j}$ for $2 \leq j \leq n_L - \ell$ in Phase 3 Round 2. If the signature $\bar{\sigma}'_L$ is not correct, then $\mathbf{V}_{L,j}$ rejects $\mathbf{S}_L$. As $\mathcal{A}$ does not know the RSA private key d_L of $\mathbf{S}_L$ and the RSA signature scheme is assumed to be secure, therefore $\mathcal{A}$ is not able to produce a valid signature $\bar{\sigma}'_L$. Therefore, $\mathcal{A}$ is not able to impersonate $\mathbf{S}_L$ to establish the new subgroup key K'_L and new final group key K' .

Next, we consider $\mathcal{A}$ wishes to impersonate subgroup controller $\mathbf{S}_i$ for $1 \leq i \neq L \leq n_c$. As $\mathbf{S}_i$ for $1 \leq i \neq L \leq n_c$ does not send any information in Phase 1 and 2, hence we only consider $\mathcal{A}$ tries to impersonate $\mathbf{S}_i$ in Phase 3. Then in Phase 3 Round 1, $\mathcal{A}$ generates $\bar{c}'_i = \mathcal{E}_{K'_i}(K' || \mathbf{S}_i)$ and $\theta'_i = \mathbf{H}(K'_i || K' || \tau_i)$. Next, $\mathcal{A}$ needs to compute valid signature $\bar{\sigma}'_i = \mathcal{S}(\mathbf{S}_i || \tau_i || \theta'_i || v)$. The validity of $\bar{\sigma}'_i$ is checked by $\mathbf{U}_{i,j}$ for $2 \leq j \leq n_i$ in Phase 3 Round 2. If the signature $\bar{\sigma}'_i$ is not correct, then $\mathbf{U}_{i,j}$ rejects $\mathbf{S}_i$. As $\mathcal{A}$ does not know the RSA private key d_i of $\mathbf{S}_i$ and the RSA signature scheme is assumed to be secure, therefore $\mathcal{A}$ is not able to produce a valid signature $\bar{\sigma}'_i$. Therefore, $\mathcal{A}$ is not able to impersonate $\mathbf{S}_i$.

Next, consider if $\mathcal{A}$ tries to impersonate remaining users $\mathbf{V}_{L,j}$ for $2 \leq j \leq n_L - \ell$. Then in Phase 1 Round 2, $\mathcal{A}$ can only try to impersonate remaining user $\mathbf{V}_{L,k} \in \mathcal{N}(\mathcal{R})$, who is a left or right neighbor of leaving users, as only $\mathbf{V}_{L,k} \in \mathcal{N}(\mathcal{R})$ sends information in Phase 1 Round 2. In this case, $\mathcal{A}$ generates $z'_{L,k} = g^{r'_{L,k}} \bmod p$ in Phase 1 Round 2 first. Next, $\mathcal{A}$ has to compute the valid signature $\sigma'_{L,k} = \mathcal{S}(\mathbf{V}_{L,k} || z'_{L,k} || \tau'_L || v)$. The validity of $\sigma'_{L,k}$ is checked by $\mathbf{S}_L$ in Phase 1 Round 3. If the signature $\sigma'_{L,k}$ is not correct, then $\mathbf{S}_L$ rejects $\mathbf{V}_{L,k} \in \mathcal{N}(\mathcal{R})$. As $\mathcal{A}$ does not know the RSA private key $d_{L,k}$ of $\mathbf{V}_{L,k} \in \mathcal{N}(\mathcal{R})$ and the RSA signature scheme is assumed to be secure, therefore $\mathcal{A}$ is not able to produce a valid signature $\sigma'_{L,k}$. Next suppose $\mathcal{A}$ tries to impersonate remaining user $\mathbf{V}_{L,j}$ for $2 \leq j \leq n_L - \ell$ in Phase 1 Round 4, then $\mathcal{A}$ generates $\kappa'_{L,j} \in \{0, 1\}^\lambda$ and $X'_{L,j} = L'_{L,j} \oplus R'_{L,j}$, where $L'_{L,j} = \mathbf{H}((z'_{L,j-1})^{r'_{L,j}} || \mathbf{SID}'_L)$, $R'_{L,j} = \mathbf{H}((z'_{L,j+1})^{r'_{L,j}} || \mathbf{SID}'_L)$, $z'_{L,n_L-\ell+1} = z'_{L,1}$, $\mathbf{SID}'_L = \mathbf{H}(\tau'_L || \mathbf{V}_{L,1} || \mathbf{V}_{L,2} || \cdots || \mathbf{V}_{L,n_L-\ell})$ and $r'_{L,j} = r_{L,j}$ and $z'_{L,j} = z_{L,j}$ if $\mathbf{V}_{L,j} \notin \mathcal{N}(\mathcal{R})$. Then $\mathcal{A}$ has to compute the valid signature $\bar{\sigma}'_{L,j} = \mathcal{S}(\mathbf{V}_{L,j} || \kappa'_{L,j} || X'_{L,j} || \tau'_L || v)$. The validity of $\bar{\sigma}'_{L,j}$ is checked by $\mathbf{S}_L$ in Phase 1 Round 5. If the signature $\bar{\sigma}'_{L,j}$ is not correct, then $\mathbf{S}_L$ rejects $\mathbf{V}_{L,j}$. As $\mathcal{A}$ does not know the RSA private key $d_{L,j}$ of remaining user $\mathbf{V}_{L,j}$ and the RSA signature scheme is assumed to be secure, therefore $\mathcal{A}$ is not able to produce a valid signature $\bar{\sigma}'_{L,j}$. Therefore, $\mathcal{A}$ is not able to impersonate remaining users $\mathbf{V}_{L,j}$ for $2 \leq j \leq n_L - \ell$.

Next, we consider $\mathcal{A}$ tries to impersonate $\mathbf{C}$. Then in Phase 2 Round 2, $\mathcal{A}$ generates $T'_L = (y'_L)^{r'_c} \bmod p$ and $T'_i = y_i^{r'_c} \bmod p$ for $1 \leq i \neq L \leq n_c$, final group key $K' = (y'_c \cdot y'_L \cdot \prod_{i=1, i \neq L}^{n_c} y_i)^{r'_c}$, $\theta'_c = \mathbf{H}(K' || y'_c)$, $c'_L = \mathcal{E}_{T'_L}(K')$ and $c'_i = \mathcal{E}_{T'_i}(K')$ for $1 \leq i \neq L \leq n_c$. Then $\mathcal{A}$ has to compute the valid signature $\sigma'_c = \mathcal{S}(\mathbf{C} || y'_c || \theta'_c)$. The validity of σ'_c is checked by $\mathbf{S}_i$ (including $\mathbf{S}_L$) for $1 \leq i \leq n_c$ in Phase 3 Round 1. If signature σ'_c is not correct, then $\mathbf{S}_i$ (including $\mathbf{S}_L$) rejects $\mathbf{C}$. As $\mathcal{A}$ does

not know the RSA private key d_C of C and the RSA signature scheme is assumed to be secure, therefore $\mathcal{A}$ is not able to produce a valid signature σ'_C . Therefore, $\mathcal{A}$ is not able to impersonate C to establish the final group key K . Hence it is not possible for $\mathcal{A}$ to obtain new subgroup key K'_j and new final group key K' in case (a).

Next we consider case (b) as follows: as the adversary $\mathcal{A}$ is not a leaving user U_{L,l_k} ($1 \leq k \leq \ell$), therefore, $\mathcal{A}$ can only obtain the public exchange information in the Leave Protocol. First we consider if $\mathcal{A}$ can obtain the new subgroup key K'_L from the public information in Phase 1. But Phase 1 of the Leave Protocol (without the signature information sent) is just the VK-GKA protocol (described in Section 9.4.1) executed by subgroup controller S_L and remaining users $V_{L,j}$ for $2 \leq j \leq n_L - \ell$ to establish the new subgroup key K'_L . By Theorem 27, it is shown that the VK-GKA protocol is secure. Therefore, the adversary $\mathcal{A}$ is not able to obtain the subgroup key K'_L from Phase 1 of the Leave Protocol.

Next, suppose the adversary $\mathcal{A}$ tries to obtain the final group key K' from Phase 2 of the Leave Protocol. But Phase 2 of the Leave Protocol (without the signature information sent) is just the VSTAR protocol (described in Section 9.4.1) where subgroup controller S_L computes and sends $y'_L = g^{K'_L}$ in Phase 2 Round 1 while S_i for $1 \leq i \neq L \leq n_c$ uses $y_i = g^{K_i}$ that C knows from previous GKA, and Phase 2 Round 2 and Phase 3 Round 1 are the same as Round 2 and Computation of Key of VSTAR protocol respectively. By Theorem 28, it is shown that the VSTAR protocol is secure. Hence $\mathcal{A}$ is not able to obtain the final group key K' from Phase 2 of the Leave Protocol. Next, we consider $\mathcal{A}$ tries to obtain the final group key K' from the public downflow information, that is, $\bar{c}'_L$ and $\bar{c}'_i$ for $1 \leq i \neq L \leq n_c$, in Phase 3 of the Leave Protocol. The new group key K' is the plaintext of $\bar{c}'_L = \mathcal{E}_{K'_L}(K' || S_L)$ and $\bar{c}'_i = \mathcal{E}_{K_i}(K' || S_i)$ for $1 \leq i \neq L \leq n_c$. As $\mathcal{A}$ does not know new subgroup key K'_L and existing subgroup keys K_i for $1 \leq i \neq L \leq n_c$, and the symmetric key encryption scheme SKE is assumed to be secure, therefore $\mathcal{A}$ is not able to obtain K' from $\bar{c}'_L$ and $\bar{c}'_i$ for $1 \leq i \neq L \leq n_c$. Hence, $\mathcal{A}$ is not able to obtain the new subgroup key K'_L and new final group key K' in the Leave Protocol. ■

Theorem 37 *Assume that the computation of RSA signature verification is efficient, then the proposed Leave protocol is resistant to DoS attack and insider attack.*

Proof: Suppose an adversary launches a DoS attack on the proposed Leave protocol which stops the GKA from producing the new final group key. In Phase 1 of the Leave Protocol, we only consider the DoS attacker launch DoS attack on subgroup controller S_L as the other subgroup controllers S_i for $1 \leq i \neq L \leq n_c$ do not send any information in Phase 1. Then in Phase 1 Round 2, S_L is able to detect the invalid signatures after verifying the signatures. As mentioned in Theorem 31, the signature verification is using RSA signature scheme, which is very fast and S_L will inform the remaining users $V_{L,j} \in \mathcal{G}'_L$ for $2 \leq j \leq n_L - \ell$ of the invalid users and the Leave protocol still continues. If the attacker tries to attack on Phase 1 Round 4, then S_L is also able to detect the invalid signature and eliminate the invalid users. Then the protocol will still continue to Phase 2. In Phase 2 of the Leave Protocol, only subgroup controller S_L sends information to main controller C in Phase 2 Round 1 to establish the new final group key while the other subgroup controllers S_i for $1 \leq i \neq L \leq n_c$ does not send any information in Phase 2. Hence if the DoS attacker tries to attack in Phase 2 Round 1, then in Phase 2 Round 2, the main controller C is able to detect and reject all invalid messages and invalid signatures sent by the DoS attacker and only accept the valid message and valid signature sent by subgroup controller S_L . Then the Leave Protocol will proceed until the new final group key is formed. Hence, the attacker will not be able to stop the proposed Leave protocol from forming the new final group key K' . Next, suppose malicious group members try to launch insider attacks to disrupt and halt the Leave Protocol by sending wrong information that causes an invalid XOR equation in subgroup $\mathcal{G}'_L$ in Phase 1 Round 5, the detection protocol described in Section 9.3 is able to detect and block these malicious group members in subgroup $\mathcal{G}'_L$. The proposed protocol will then continue to establish the new subgroup key K'_L and the new final group key K' , thus preventing the insider attack. The malicious group members are not able to launch insider attacks in subgroups $\mathcal{G}_i \neq \mathcal{G}'_L$ for $1 \leq i \neq L \leq n_c$ in the Leave Protocol as subgroups $\mathcal{G}_i \neq \mathcal{G}'_L$ do not send and receive any information in Phase 1 of the Leave Protocol. Hence, the Leave protocol is able to resist DoS attack and insider attack. ■

9.5 Complexity Analysis

In Table 9.1, we analyze and compare the computational (number of modular exponentiations performed, signature generation and verification and certificate verification) and communication

(number of messages sent and received) costs of the proposed Initial GKA, Join and Leave Protocol with the Initial GKA, Join and Leave Protocols from two other GKA protocols, namely Kim's GKA protocol [KLL04] and NLKKW protocol [NLK⁺04]. We compare with the NLKKW protocol [NLK⁺04] instead of the NLKW protocol [NLKW05] because the NLKW protocol [NLKW05] does not provide Join and Leave Protocol. Besides, the NLKKW protocol [NLK⁺04] is similar to the NLKW protocol [NLKW05] and the main difference is that the NLKKW protocol [NLK⁺04] is based on the difficulty of the factoring problem while the NLKW protocol [NLKW05] is based on the difficulty of the DDH problem. The terms m and ℓ represents the number of joining users and number of leaving users respectively. In Kim's GKA Protocol, U_i for $1 \leq i \leq n$ refers to users in the group where n is the group size, $U_j \in \mathcal{J}$ for $1 \leq j \leq m$ refers the number of joining users, U_w refers to the left and right neighbors of leaving users and $U_{n''}$ refers to the last user in the group, where n'' refers to the group size less the leaving users. In the NLKKW protocol, $U_i \neq \mathcal{C}$ refers to users in the group, $\mathcal{C}$ denotes the controller node, $U_j \in \mathcal{J}$ for $1 \leq j \leq m$ refers the number of joining users and $V_i \neq \mathcal{C}$ denotes the remaining users in the group after a user leave event.

Initial Group Key Agreement

From Tables 9.1, it can be seen that the number of signature and certificate verifications as well as the number of messages received by users $U_{i,j}$ for $1 \leq i \leq n_c$ and $2 \leq j \leq n_i$ in the proposed Initial GKA protocol is much lower as compared to user nodes in Kim's GKA protocol [KLL04]. The users U_i in NLKKW appears to be more efficient than the proposed Initial GKA protocol, however, the single message received by each U_i in NLKKW is a very large message containing $2 \times \mathcal{O}(n)$ 1024-bits information for each U_i to compute the final group key. Furthermore, the number of exponentiations, signature and certificate verifications and number of messages received by the main controller $\mathcal{C}$ in NLKKW is $\mathcal{O}(n)$, which is much higher as compared to $\mathcal{O}(n_c)$ and $\mathcal{O}(n_i)$ by the main controller $\mathcal{C}$ and subgroup controllers S_i for $1 \leq i \leq n_c$ respectively in the proposed Initial GKA protocol.

Join Protocol

Existing users $U_{J,j} \in \mathcal{G}_J$ for $2 \leq j \leq n_J - 1$ and users $U_{i,j}$ for $1 \leq i \neq J \leq n_c$ and $2 \leq j \leq n_i$ in the proposed Join Protocol need not perform modular exponentiations while all nodes in Kim's Join

Table 9.1: Computational and Communication Costs

			Exp.	Sign Gen	Sign Verif	Cert Verif	Msgs Sent	Msgs Received
I G K A	Kim's [KLL04]	U_i	3	2	$n + 1$	$n - 1$	2	$n + 1$
		C	$n + 1$	1	$n - 1$	$n - 1$	1	$n - 1$
	NLKKW [NLK ⁺ 04]	$U_i \neq C$	2	1	1	1	1	1
		C	$n_c + 2$	1	n_c	n_c	1	n_c
	Proposed IKA Protocol	S_i	5	4	$2n_i + 1$	$n_i + 1$	5	$2n_i + 1$
		$U_{i,j}$	3	2	3	1	2	4
J O I N	Kim's [KLL04]	U_n	3	2	$2m + 4$	m	2	$2m + 4$
		U_1, U_2	3	2	$2m + 4$	m	2	$2m + 4$
		$U_3 - U_{n-1}$	2	—	$2m + 5$	m	—	$2m + 5$
		$U_j \in \mathcal{J}$	3	2	$2m + 4$	$m + 2$	2	$2m + 4$
	NLKKW [NLK ⁺ 04]	C	$n + m + 1$	1	m	m	1	m
		$U_i \neq C$	1	—	1	—	—	1
		$U_j \in \mathcal{J}$	2	1	1	1	1	1
	Proposed Join Protocol	C	$n_c + 2$	1	1	—	1	1
		S_J	5	4	$2m + 3$	m	5	$2m + 3$
		$S_i \neq S_J$	1	1	1	—	1	1
		$U_{J,k} \in \mathcal{J}$	3	2	3	1	2	4
		$U_{J,n_J} \in \mathcal{G}_J$	3	2	3	0	2	4
		$U_{J,j} \in \mathcal{G}_J$	—	—	1	—	—	2
		$U_{i,j}$ for $i \neq J$	—	—	1	—	—	1
L E A V E	Kim's [KLL04]	$U_{n''}$	2	2	$n + \ell$	—	2	$n + \ell$
		$U_w \neq U_{n''}$	2	2	$n + \ell$	—	2	$n + \ell$
		$U_i \neq (U_w, U_{n''})$	—	1	$n + \ell$	—	1	$n + \ell$
	NLKKW [NLK ⁺ 04]	C	$n - \ell + 1$	1	—	—	1	—
		$V_i \neq C$	1	—	1	—	—	1
	Proposed Leave Protocol	C	$n_c + 2$	1	1	—	1	1
		S_L	$5(2)$	4	$n_L + \ell + 1$	—	5	$n_L + \ell + 1$
		$S_i \neq S_L$	1	1	1	—	1	1
		$V_{L,k} \in \mathcal{N}(\mathcal{R})$	3	2	3	—	2	4
		$V_{L,j} \notin \mathcal{N}(\mathcal{R})$	$1(0)$	1	3	—	1	4
		$U_{i,j}$ for $i \neq L$	—	—	1	—	—	1

Note: S_L performs 5 mod. exp. if $S_L \in \mathcal{N}(\mathcal{R})$, otherwise S_L performs 2 mod. exp. if $S_L \notin \mathcal{N}(\mathcal{R})$. Only $V_{L,j} \notin \mathcal{N}(\mathcal{R})$, who are left or right neighbor of $V_{L,k} \in \mathcal{N}(\mathcal{R})$, performs 1 exp., otherwise, $V_{L,j} \notin \mathcal{N}(\mathcal{R})$ do not need to perform any exp.

Protocol and NLKKW Join Protocol need to perform modular exponentiations. Furthermore, $U_{J,j}$ for $2 \leq j \leq n_J - 1$ only has to receive 2 messages and verify 1 message while $U_{i,j}$ for $1 \leq i \neq J \leq n_c$ and $2 \leq j \leq n_i$ only has to receive and verify 1 message. In comparison, U_1, U_2 and U_n in Kim's Join protocol has to perform 3 exponentiations, receive $2m + 4$ messages, verify $2m + 4$ signatures and

verify m certificates while users $U_3 - U_{n-1}$ in Kim's Join protocol has to perform 2 exponentiations, receive $2m + 5$ messages, verify $2m + 5$ signatures and verify m certificates. The new joining users $U_j \in \mathcal{J}$ in Kim's Join protocol has to compute 3 exponentiations, receive $2m + 4$ messages, verify $2m + 4$ signatures and verify $m + 2$ certificates while new joining users $U_{J,k}$ for $n_J + 1 \leq k \leq n_J + m$ in the proposed Join Protocol have to perform 3 exponentiations, receive 4 messages, verify 3 signatures and verify 1 certificate regardless of the number of joining users m . The existing last user U_{J,n_J} performs 3 exponentiations, receives 4 messages and verifies 3 signatures in the Join Protocol. Although $U_j \in \mathcal{J}$ in NLKKW Join Protocol only has to perform 2 exponentiations and receive 1 message, however the received message contains $2 \times \mathcal{O}(n + m)$ 1024-bits information for all users (both new and existing users) to compute the new final group key. This large message is also received by existing users $U_i \neq C$ in NLKKW Join Protocol. Controller C in NLKKW Join Protocol has to perform $n + m + 1$ exponentiations, which is the most number of exponentiations while in comparison, main controller C , subgroup controller S_J and S_i for $1 \leq i \neq J \leq n_c$ in the proposed Join Protocol only has to perform $n_c + 2$, 5 and 1 exponentiations respectively, regardless of group size n and number of joining users m .

Leave Protocol

Users $U_{i,j}$ for $1 \leq i \neq L \leq n_c$ and $2 \leq j \leq n_i$ in the proposed Leave Protocol do not have to perform any exponentiations while only $V_{L,j} \notin \mathcal{N}(\mathcal{R})$, who are left or right neighbors of $V_{L,k} \in \mathcal{N}(\mathcal{R})$, performs 1 exponentiation and all other $V_{L,j} \notin \mathcal{N}(\mathcal{R})$ do not need to perform any exponentiations. Although $U_i \neq (U_w, U_{n''})$ in Kim's Leave protocol also do not have to perform any exponentiations, however $U_i \neq (U_w, U_{n''})$ have to receive $n + \ell$ messages and verify $n + \ell$ signatures, which is not scalable as it depends on the group size n . Similarly, $U_{n''}$ and $U_w \neq U_{n''}$ of Kim's Leave protocol also has to receive $n + \ell$ messages and verify $n + \ell$ signatures, which makes Kim's Leave protocol rather inefficient. Although S_L in the proposed Leave Protocol has to perform 5 exponentiations if $S_L \in \mathcal{N}(\mathcal{R})$ (or 2 exponentiations if $S_L \notin \mathcal{N}(\mathcal{R})$), S_i for $1 \leq i \neq L \leq n_c$ has to perform 1 exponentiation and main controller C has to perform $n_c + 2$ exponentiations, however controller C in NLKKW Leave Protocol has to compute $n - \ell + 1$ exponentiations, which is large if group size n is large.

9.6 Energy Analysis

In this section, we perform the computational and communication energy cost analysis of user nodes in the proposed GKA protocols and two other GKA protocols, namely Kim's GKA protocol and NLKKW protocol. We do not show the energy cost analysis for main controller and subgroup controllers as they are high power and less energy constrained nodes.

As the 206MHz SA-1110 "StrongARM" microprocessor are typically used in user devices such as PDAs for computation of cryptographic operations while the Lucent IEEE 802.11 11Mbps WaveLAN card is used as a wireless network interface in user devices, hence our computational energy costs are based on the 206MHz "StrongARM" microprocessor as discussed in Section 3.3.2 while our communication energy costs are based on the Lucent WaveLAN card discussed in Section 3.4.2 respectively (i.e. for the energy analysis, we assume that the user nodes are devices such as PDAs that run on the 206MHz "StrongARM" microprocessor and uses the Lucent IEEE 802.11 11Mbps WaveLAN card to communicate on the wireless medium). In Section 9.6.1, we provide the energy cost comparison of user nodes running the proposed Initial GKA, Join and Leave Protocol with user nodes running the Initial GKA, Join and Leave Protocol of the other two GKA protocols.

9.6.1 Energy Consumption Cost Comparison Results

In this section, we first provide a chart (Figure 9.1) which compares the energy costs of user nodes in the proposed Initial GKA protocol and the Initial GKA protocol of two other protocols, namely Kim's GKA protocol and NLKKW protocol, for different number of user nodes (1000, 2500, 7500 and 10000 respectively). For our proposed Initial GKA protocol, the number of subgroups $n_c = 10$ while the number of user nodes in each subgroup (excluding subgroup controller S_i) are 100, 250, 750 and 1000 for total number of user nodes 1000, 2500, 7500 and 10000 respectively. Next, we provide two tables, namely Table 9.2 and 9.3, to compare the energy costs of user nodes involved in the Join and Leave Protocols respectively. Table 9.2 shows the energy costs of various user nodes in the proposed Join Protocol and the Join Protocol of the other two GKA protocols, namely Kim's GKA protocol and NLKKW protocol for $m = 20$ new users joining a group of $n = 7500$ nodes. Table 9.3 shows the energy costs of various user nodes in the proposed Leave Protocol and the Leave Protocol of the other two GKA protocols, namely Kim's GKA protocol and NLKKW protocol for $\ell = 20$

users leaving a group with $n = 7500$ nodes. It should be noted that the number of user nodes in each subgroup is 750 in our proposed Join and Leave Protocol since the group size $n = 7500$.

Initial Group Key Agreement

Based on the complexity analysis in Table 9.1 and the energy costs in Tables 3.3 and 3.5 in Chapter 3, we obtain the total energy costs of each user node running Kim's Initial GKA protocol, NLKKW Initial GKA protocol and the proposed Initial GKA protocol respectively in Figure 9.1. The Y-axis in the figure measures the energy consumed in milliJoules (mJ) while the X-axis shows the total number of user nodes (1000, 2500, 7500 and 10000 respectively). The numbers shown inside the boxes in Figure 9.1 shows the total energy costs (in mJ) for each user node in Kim's Initial GKA protocol, NLKKW Initial GKA protocol and the proposed Initial GKA protocol respectively. For example, when the number of user nodes is 7500, the users $U_{i,j}$ for $1 \leq i \leq n_c$ and $2 \leq j \leq n_i$ in the proposed Initial GKA protocol consumes only $277.55mJ$ as compared to $545.12mJ$ by the user nodes U_i in NLKKW Initial GKA protocol and $13226.98mJ$ consumed by users U_i in Kim's Initial GKA protocol.

Figure 9.1: Energy Consumption Cost of user nodes for Initial GKA protocol.

From Figure 9.1, it can be seen that even though the proposed Initial GKA protocol is the only GKA protocol to provide both key confirmation and protection against insider and DoS attacks, the users $U_{i,j}$ for $1 \leq i \leq n_c$ and $2 \leq j \leq n_i$ in the proposed Initial GKA protocol are the most

energy efficient for different number of user nodes. Furthermore, the proposed protocol is also the most scalable due to the use of subgroups in the Central-ContRi group model which helps to reduce the number of computations and messages received by users $U_{i,j}$ despite the large group size. For users U_i in Kim's GKA protocol, the two main factors for high energy costs are the large amount of computation involved (signature and certification verifications) and the large number of messages received from other user nodes. The main reason for high energy costs for users U_i in NLKKW is the very large message received from controller C that contains information for all users U_i to compute the group key.

Join Protocol

Table 9.2 shows the total energy costs of different user nodes when running Kim's Join protocol, NLKKW Join protocol and the proposed Join Protocol respectively. These results are based on the complexity analysis in Table 9.1 and the energy costs in Tables 3.3 and 3.5 in Chapter 3. We assume that the current group size $n = 7500$ and there are $m = 20$ new users joining the group. It should be noted that the number of user nodes in each subgroup $\mathcal{G}_i$ in our proposed Join Protocol is $n_i = 750$ since group size $n = 7500$.

Table 9.2: User Energy Costs for Join Protocol(20 Users)

		Energy Consumed (mJ)
Kim's GKA [KLL04]	U_n	120.82
	U_1, U_2	120.78
	$U_3 - U_{n-1}$	77.62
	$U_j \in \mathcal{J}$	122.38
NLKKW [NLK ⁺ 04]	$U_i \neq C$	519.09
	$U_j \in \mathcal{J}$	546.5
Proposed Join Protocol	$U_{J,k} \in \mathcal{J}$	67.34
	$U_{J,n_J} \in \mathcal{G}_J$	66
	$U_{J,j} \in \mathcal{G}_J$	1.02
	$U_{i,j}$ for $i \neq J$	0.92

From Table 9.2, it can be seen that the proposed Join Protocol is the most energy efficient. Existing users $U_{J,j}$ for $2 \leq j \leq n_J - 1$ and $U_{i,j}$ for $1 \leq i \neq J \leq n_c$ and $2 \leq j \leq n_i$ only consumes $1.02mJ$ and $0.92mJ$ respectively, while new users $U_{J,k}$ for $n_J + 1 \leq k \leq n_J + m$ and existing last user U_{J,n_J} in the proposed Join Protocol consumes $67.34mJ$ and $66mJ$ respectively, which is still

lower than all users in Kim's Join Protocol and NLKKW Join Protocol as shown in Table 9.2. The main reason for the high energy consumption by $U_i \neq C$ and $U_j \in \mathcal{J}$ in NLKKW Join Protocol is due to the very large message received from the controller C . The main reason for energy cost of users U_i for $1 \leq i \leq n + m$ (including new users) in Kim's Join Protocol is due to the high computational costs (exponentiation, signature and certificate verifications).

Leave Protocol

Table 9.3 shows the total energy costs of different user nodes when running Kim's Leave protocol, NLKKW Leave protocol and the proposed Leave Protocol respectively. These results are based on the complexity analysis in Table 9.1 and the energy costs in Tables 3.3 and 3.5 in Chapter 3. Again, we assume that the current group size $n = 7500$ and there are $\ell = 20$ users leaving the group. It should be noted that the number of user nodes in each subgroup $\mathcal{G}_i$ in our proposed Leave Protocol is $n_i = 750$ since group size $n = 7500$.

Table 9.3: User Energy Costs for Leave Protocol (20 Users)

		Energy Consumed (mJ)
Kim's GKA [KLL04]	$U_{n''}$	6713.8
	$U_w \neq U_{n''}$	6713.76
	$U_i \neq (U_w, U_{n''})$	6677.18
NLKKW [NLK ⁺ 04]	$V_i \neq C$	267.42
Proposed Leave Protocol	$V_{L,k} \in \mathcal{N}(\mathcal{R})$	65.8
	$V_{L,j} \notin \mathcal{N}(\mathcal{R})$	29.34/19.55
	$U_{i,j}$ for $i \neq L$	0.92

Note: Only $V_{L,j} \notin \mathcal{N}(\mathcal{R})$, who are left or right neighbor of $V_{L,k} \in \mathcal{N}(\mathcal{R})$, consumes $29.34mJ$. Otherwise, $V_{L,j} \notin \mathcal{N}(\mathcal{R})$ consumes $19.55mJ$.

From Table 9.3, it can be seen that the proposed Leave Protocol is the most energy efficient since $V_{L,k} \in \mathcal{N}(\mathcal{R})$, $V_{L,j} \notin \mathcal{N}(\mathcal{R})$ for $2 \leq j \leq n_L - \ell$ and $U_{i,j}$ for $1 \leq i \neq L \leq n_c$ and $2 \leq j \leq n_i$ consumes only $65.8mJ$, $29.34mJ/19.55mJ$ and $0.92mJ$ respectively. Users in Kim's Leave protocol are the least efficient due to the large number of signature verifications ($n - \ell$ signature verifications) and the large number of received messages ($n - \ell$ messages). The main reason for the high energy consumption for users $V_i \neq C$ in NLKKW Leave Protocol is due to the large message to be received from the controller C .

9.7 Summary

GKA protocols are vulnerable to DoS attacks and most current GKA protocols do not consider protection against DoS attacks. Besides outsider attacks, the threat of malicious group members launching attacks to disrupt GKA service in heterogeneous networks is also very serious and it is therefore important to ensure the availability of GKA services to provide secure group communications in heterogeneous networks. However, most current GKA protocols focus only on security against outsiders. In this chapter, we proposed three authenticated, energy efficient and scalable GKA protocols, namely the Initial GKA, Join and Leave Protocols that provide both key confirmation and protection against DoS attacks and insider attacks. We also proposed a detection protocol that can detect and block malicious group members and continue to establish a subgroup key and final group key without requiring to halt the GKA protocol, thus preventing the attack launched by malicious group members. A complexity and energy analysis indicates that the proposed Initial GKA, Join and Leave Protocols are energy efficient, scalable and suitable for use in heterogeneous networks.

Chapter 10

Denial-of-Service Password-based GKA for Wireless Networks

10.1 Introduction

In this chapter, we shall focus on asymmetric wireless networks (Wi-Fi and WMNs) where there are high power devices (access points and mesh routers) while the user nodes (PDAs and laptop computers) are typically battery-operated low power devices.

The current technique to provide authentication in GKA protocols through the use of digital signatures schemes such as DSA require a Public Key Infrastructure (PKI) and also require energy-intensive computations such as modular exponentiations. Furthermore, there may be an absence of PKI in some wireless networks. On the other hand, password-based GKA protocols provide better authentication solutions since password-based authentication do not require PKI and expensive computational operations.

One important security requirement in secure group communications is the availability of GKA services to establish a common group key without disruptions. However, GKA protocols are susceptible to Denial-of-Service (DoS) attacks and most current GKA protocols, including password-based GKA protocols, such as [ABCP06, BCP02b, BLLH06, DB05, KJL06, LHL04, NKW05, NLK⁺04] are not designed to protect against DoS attacks. For example, in a recent password-based GKA protocol, PAGKE [KJL06], an adversary can simply launch a DoS attack by sending invalid messages in Round 1 or Round 2 without knowing the password pw used. The honest user nodes will still

perform energy-intensive modular exponentiations before discovering the invalid messages. The protocol will then halt and be re-started again. However, the adversary can simply continue to send invalid messages in every re-started GKA protocol run and result in the failure of the GKA services as no final group key will be established. Besides password-based GKA protocols, most two party authentication and key exchange protocols are also susceptible to DoS attacks that disrupts the authentication and key exchange services provided.

Besides DoS attacks, password-based GKA protocols are also susceptible to off-line and on-line password guessing attacks where the attacker may recover the password and use it to impersonate the legitimate user. In off-line password guessing attacks, an attacker obtains a set of information, then he/she guesses a password and verify its correctness in an off-line manner. While in on-line password guessing attacks, an attacker guesses a password and interactively participates in the protocol run to verify the correctness of the password. Although most current password-based GKA protocols are secure against off-line password guessing attacks, they are not design to protect against on-line password guessing and DoS attacks [ABCP06,BLLH06,Dut07].

This chapter consist of two parts. In the first part of this chapter, we show that Wan et al's Authentication and Key Exchange Protocol [WDBA05] described in Section 2.5.3 of Chapter 2 is not DoS-resistant, that is, an adversary can launch three different DoS attacks against the authentication server in their protocol. We then propose a new DoS-Resistant authentication and key exchange protocol that is free from the flaws in Wan et al's authentication and key exchange protocol. Our proposed authentication and key exchange protocol provides both client anonymity and protection against DoS attacks and other desirable security requirements such as mutual authentication, key confirmation and security against dictionary attacks. Our proposed authentication and key exchange protocol also uses password-based authentication. We then provide a security and complexity analysis of our proposed authentication and key exchange protocol and show that our proposed authentication and key exchange protocol is both secure and efficient for authentication and key exchange in wireless networks.

In the second part of this chapter, we proposed authenticated password-based Initial GKA protocol, Join Protocol and Leave Protocol that provide protection against DoS attacks and is secure against both off-line and on-line password guessing attacks. Besides providing key confirma-

tion, the proposed protocols, which use the Centralized-Centralized (Central-Central) group model (described in Section 2.12.4 of Chapter 2), are also energy efficient and scalable. Although we have proposed another GKA protocol against DoS attacks in Chapter 9, however there are several differences between the proposed protocol in this chapter and the GKA protocol in Chapter 9. For example, the proposed protocol in this chapter uses password-based authentication whereas the GKA protocol in Chapter 9 uses digital signature for authentication. Furthermore, the GKA protocol in Chapter 9 uses a different hierarchical group structure as compared to the proposed protocol in this paper. Besides, the GKA protocol described in Chapter 9 is also different from the proposed protocol described in this chapter.

Our proposed password-based GKA protocols in this chapter uses different passwords shared between the controllers and users as compared to a common password shared between users or between users and server/controller. If a common password is shared by all users, then the password has to be changed and updated securely if any user lost the password. Furthermore, users leaving the group will still know the common password and this makes it difficult to securely update all remaining users with the new password. Although Byun et al's EKE-D protocol in [BLLH06] also uses different passwords, their protocol does not provide protection against DoS attacks and on-line password guessing attacks. Our password-based GKA protocol provides a better solution against DoS attacks as the nodes need not perform computationally intensive digital signature verifications on invalid messages sent by attackers. This is because the proposed password-based GKA protocol uses symmetric key encryption, message authentication code and hash function which have very low computation time as compare to modular exponentiations used in digital signature schemes (e.g. DSA). We perform a complexity and energy analysis of the proposed password-based Initial GKA protocol with two current password-based GKA protocols, namely Abdalla et al's protocol [ABCP06] and Byun et al's EKE-D protocol [BLLH06], and show that even though the proposed protocol provides protection against DoS attacks and is secure against both on-line and off-line password guessing attacks, the proposed protocol is still more energy efficient and scalable.

10.2 Security Analysis and Improvement of a DoS-Resistant Authentication and Key Exchange Protocol

Wireless networks can be set up in various public places such as airports, hotels and shopping malls and only authorized users (legitimate clients) are allowed to access the wireless network services. Hence, there is a need to provide secure authentication and key exchange mechanism to prevent unauthorized users from gaining access to the network. Privacy problems can occur in wireless networks as an adversary can use the knowledge of the client's identity to locate the client and track down his/her movement. Hence, confidentiality protection of the client's identity is important.

Password-based authenticated key exchange (PAKE) protocols provide good solutions for authentication and key exchange due to the user friendliness of human memorable passwords. However, most PAKE protocols do not provide client anonymity and DoS-resistance for wireless networks. In 2001, Schmid et al [SFW⁺01] proposed an access control protocol known as the Lancaster protocol which is also considered as an authentication and key exchange protocol for accessing the wireless networks. The Lancaster protocol uses password-based authentication and provides client anonymity. However, Wan et al [WDBA05] pointed out the security weaknesses and design flaws in the Lancaster protocol and proposed a DoS-Resistant anonymous authentication and key exchange protocol, which also uses passwords. In the next section, we show three different DoS attacks against the authentication server in Wan et al's Authentication and Key Exchange Protocol described in Section 2.5.3 of Chapter 2. Next, we propose a new DoS-Resistant authentication and key exchange protocol that is free from the flaws in Wan et al's protocol. A security and complexity analysis show that our proposed protocol is both secure and efficient for authentication and key exchange in wireless networks.

10.2.1 Attacks on Wan et al's Authentication and Key Exchange Protocol

We show three different DoS attacks against the authentication server in Wan et al's Authentication and Key Exchange Protocol [WDBA05] as follows:

Let $\mathcal{A}$ be the adversary that tries to launch a DoS attack against the authentication server S in Wan et al's DoS-Resistant authentication and key exchange protocol. In this section, we show

that $\mathcal{A}$ can launch three different DoS attacks against Wan et al's authentication and key exchange protocol to deplete S 's computation and storage resources.

Attack 1

In this attack, $\mathcal{A}$ launches the DoS attack against S by sending invalid messages in Round 1 of Wan et al's authentication and key exchange protocol. The attack is as follows:

Step 1: In Round 1 of Wan et al's authentication and key exchange protocol, $\mathcal{A}$ selects a random $\pi \in Z_p$ as the encrypted message (i.e. $z_c^* = \pi$) and sends π to S .

As S does not know that $z_c^* = \pi$ received from $\mathcal{A}$ is invalid in Round 2, S will continue with the protocol by first selecting random $y \in Z_q^*$ and computing $z_s = g^y \bmod p$ and hash value $\bar{v} = H_{HK}(z_s, \pi)$. S then sends $\{S, z_s, \pi, \bar{v}\}$ to $\mathcal{A}$.

Step 2: $\mathcal{A}$ does not respond to S and launches a new Wan et al's authentication and key exchange protocol with S by repeating Step 1 and 2 using another random $\pi \in Z_p$ as the encrypted message to be sent to S . $\mathcal{A}$ continues this process to deplete S 's system resources.

Due to this attack, S will keep selecting random $y \in Z_q^*$ and compute $z_s = g^y \bmod p$ in Round 2 for every protocol run with the $\mathcal{A}$. This will cause S to continuously perform computationally expensive modular exponentiation operations, which will deplete S 's computation resources, thus allowing $\mathcal{A}$ to succeed in the DoS attack. Although Wan et al mentioned that S can generate the random exponential z_s beforehand or periodically and also re-use previously used z_s , however, this would mean that S has to keep storing the periodically generated z_s and also store previously generated z_s to be re-used since $\mathcal{A}$ keeps sending different π as different encrypted messages z_c^* to S . This results in S 's storage resources to be depleted, thus allowing $\mathcal{A}$ to succeed in the DoS attack.

Attack 2

In Wan et al's authentication and key exchange protocol, the authors did not specify the public key encryption scheme PKE to be used to encrypt/decrypt ciphertext $c_s = E_{p\kappa}(C, \text{MAC}_c, N_c, \theta_c)$. In this attack, we show that $\mathcal{A}$ can launch a DoS attack against S in Round 3 of Wan et al's authentication and key exchange protocol, assuming that the encryption operation of the PKE used (e.g RSA encryption) is efficient. The RSA [RSA78] scheme is a likely candidate for use as the PKE scheme in Wan et al's authentication and key exchange protocol as it is popular and widely available. RSA encryption (with $e = 65537$) only takes $0.22ms$ using the MIRACL Library [Mir]

on a 3GHz Pentium IV system. Hence, RSA encryption is efficient. $\mathcal{A}$ pretends to be legitimate C and launches the attack as follows:

Step 1: This step is similar to Step 1 of Attack 1.

Step 2: After receiving $\{S, z_s, \pi, \bar{\nu}\}$ from S in Round 2, $\mathcal{A}$ randomly selects $\bar{N}_c, \bar{\theta}_c \in Z_q^*$, encrypts $\{C, \text{MAC}_C, \bar{N}_c, \bar{\theta}_c\}$ using the S 's public key $p\kappa$ and obtains $\bar{c}_s = E_{p\kappa}(C, \text{MAC}_C, \bar{N}_c, \bar{\theta}_c)$. Finally, $\mathcal{A}$ sends $\{z_s, \pi, \bar{\nu}, \bar{c}_s\}$ to S .

Upon receiving the above message from $\mathcal{A}$, S will verify that the hash value $\bar{\nu}$ is correct because $\mathcal{A}$ did not alter $\bar{\nu}$. Next, S decrypts $\bar{c}_s$ and fetches C 's password pw that $\mathcal{A}$ is masquerading. S then decrypts π and obtains some random $\hat{\pi}$. As S does not know that $\hat{\pi}$ is invalid yet, S will continue to perform computationally expensive modular exponentiation operation to compute session key $k = H(C, S, \hat{\pi}^y)$. S will only realize the invalid $\hat{\pi}$ after verifying that $\bar{\theta}_c \neq H_k(z_s, \pi)$.

Step 3: $\mathcal{A}$ does not respond to S and launches a new Wan et al's authentication and key exchange protocol with S by repeating Step 1 to 3 using different random π in Round 1 and different randoms $\bar{N}_c$ and $\bar{\theta}_c$ in Round 3. $\mathcal{A}$ repeats this process to deplete the system resources of S .

Although S can detect the invalid $\hat{\pi}$ sent by $\mathcal{A}$ in Step 2, but S has to perform computationally expensive modular exponentiation operations first. As $\mathcal{A}$ executes many instances of Wan et al's authentication and key exchange protocol with S , S 's computation resources would be depleted due to the large number of computationally expensive modular exponentiation operations required. Hence $\mathcal{A}$ is able to succeed in this DoS attack.

Attack 3

In this attack, we show that $\mathcal{A}$ can launch another DoS attack against S , assuming that the decryption operation of the PKE used (e.g RSA decryption) is computationally expensive. As mentioned in Attack 2, the popular and widely available RSA scheme is likely to be used as the PKE scheme in Wan et al's authentication and key exchange protocol. RSA decryption takes $3.88ms$ as compared to $0.22ms$ for RSA encryption using the MIRACL Library [Mir] on a 3GHz Pentium IV system. Hence, RSA decryption is computationally expensive. $\mathcal{A}$ launches the attack as follows:

Step 1: This step is similar to Step 1 of Attack 1 and 2.

Step 2: After receiving $\{S, z_s, \pi, \bar{\nu}\}$ from S in Round 2, $\mathcal{A}$ randomly selects a string $\tilde{c}_s$ that is the same size as encrypted message c_s in Round 3 and sends $\{z_s, \pi, \bar{\nu}, \tilde{c}_s\}$ to S .

Step 3: $\mathcal{A}$ does not respond to S and launches a new Wan et al's authentication and key exchange protocol with S by repeating Step 1 to 3 using different random π in Round 1 and different random string $\tilde{c}_s$ in Round 3. $\mathcal{A}$ repeats this process to deplete the system resources of S .

After receiving $\{z_s, \pi, \bar{\nu}, \tilde{c}_s\}$ in Step 2, S will verify that $\bar{\nu}$ is correct and perform computationally expensive public key decryption (e.g. RSA decryption) to decrypt $\tilde{c}_s$ even though $\tilde{c}_s$ is a random string (invalid ciphertext). As $\mathcal{A}$ executes many instances of Wan et al's authentication and key exchange protocol with S , S would have to perform a large number of computationally expensive public key decryption (e.g. RSA decryption) operations and deplete its computation resources, thus allowing $\mathcal{A}$ to succeed in the DoS attack.

10.2.2 Proposed Authentication and Key Exchange Protocol

In this section, we present our proposed authentication and key exchange protocol that avoids the flaws in Wan et al's authentication and key exchange protocol and provides both client anonymity and protection against DoS attacks. We assume that the client C shares a password pw with the authentication server S . The password pw is of low entropy and drawn uniformly at random from the dictionary Password of size $\hat{N}$. In addition, we assume that C and S shares a pairwise secret key κ that is securely preloaded into the tamper-proof devices in C and S during the setup phase. Next, two random large primes, q and p , are selected such that q divides $p - 1$. (p is 1024 bits and q is 160 bits) and an element $g \in Z_p^*$ of order q is selected.

The proposed authentication and key exchange protocol consist of six rounds. In Round 1, C starts the protocol by sending a HELLO message to S . In Round 2, S selects a random nonce N_s of L -bits long and computes a hash value $\lambda = H_1(S, N_s)$. S then splits N_s into two strings, namely $N_s\langle 1, b \rangle$, which is the first b bits of N_s , and $N_s\langle b + 1, L \rangle$, which is the $(b + 1)^{\text{th}}$ to L^{th} bits of N_s . S will only send $N_s\langle b + 1, L \rangle$ to C , thus requiring C to compute $N_s\langle 1, b \rangle$ using brute force to obtain hash value λ . The difficulty of the brute force computation of $N_s\langle 1, b \rangle$ depends on the size of b . The value of b is typically set so that the computation time for $N_s\langle 1, b \rangle$ and λ is equal to or slightly longer than the computation time of a public key decryption operation, which the server S has to perform in Round 4. In Round 4, S first checks if C has responded before the expiry time and also if C has computed $N_s\langle 1, b \rangle$ correctly. If yes, then S and C will authenticate each other and establish

a session key k in Rounds 4 and 5. The session key k will then be sent securely to the access point AP by S in Round 6. C and AP can then use k for secure data communications in the network. The proposed authentication and key exchange protocol works as follows:

Round 1: C initiates the protocol by sending a HELLO message to S via AP.

Round 2: S first selects a L -bits long random nonce N_s and computes the hash value $\lambda = H_1(S, N_s)$. Next, S splits N_s into two strings, namely $N_s\langle 1, b \rangle$ and $N_s\langle b+1, L \rangle$. Finally S sends $\{S, N_s\langle b+1, L \rangle, \lambda, b\}$ to C and sets an expiry time for C to respond.

Round 3: C computes by brute force the b -bits string $N'_s\langle 1, b \rangle$ which satisfies the hash equation $\lambda = H_1(S, N'_s\langle 1, b \rangle, N_s\langle b+1, L \rangle)$. Next, C selects a L -bits random nonce N_c , random $x \in Z_q^*$ and computes $z_c = g^x \bmod p$ and $\mu_c = \mathcal{M}_\kappa(z_c, N_c, N_s, C, S, 00)$, where κ is the secret key shared between C and S. C then encrypts $\{C, \text{MAC}_C, \text{pw}, S, N_c, N'_s\langle 1, b \rangle, N_s\langle b+1, L \rangle, \mu_c\}$ using S's public key $p\kappa$ and obtains ciphertext $c_s = E_{p\kappa}(C, \text{MAC}_C, \text{pw}, S, N_c, N'_s\langle 1, b \rangle, N_s\langle b+1, L \rangle, \mu_c)$. Finally, C sends $\{z_c, N'_s\langle 1, b \rangle, N_s\langle b+1, L \rangle, c_s\}$ to S.

Round 4: S first checks if C has responded before the expiry time. If the expiry time has past, S ignores C. Otherwise, S checks if C has computed the correct $N_s\langle 1, b \rangle$. If $N_s\langle 1, b \rangle \neq N'_s\langle 1, b \rangle$, S blocks malicious C. Otherwise, S uses its private key $s\kappa$ to decrypt ciphertext c_s and check if $C, \text{MAC}_C, S, \text{pw}$ and N_s in c_s are decrypted correctly. If password pw is incorrect, S blocks malicious C. Otherwise, S obtains N_c and μ_c . Next, S uses κ to verify μ_c . If μ_c is incorrect, S blocks malicious C. Otherwise, S selects random $y \in Z_q^*$ and computes $z_s = g^y \bmod p$, session key $k = H_2(C, S, N_c, N_s, z_c^y)$, key confirmation code $\theta_s = H_1(S, C, N_s, N_c, k, z_s, z_c, 01)$ and $\mu_s = \mathcal{M}_\kappa(z_s, \theta_s, N_s, N_c, S, C, 10)$. S then encrypts $\{C, S, \text{IP}_C, N_c, N_s\}$ using session key k and obtains $c'_s = \mathcal{E}_k(C, S, \text{IP}_C, N_c, N_s)$ and sends $\{S, z_s, \theta_s, \mu_s, c'_s\}$ to C.

Round 5: C checks if μ_s is valid. If μ_s is not valid, C rejects S. Otherwise, C computes session key $k = H_2(C, S, N_c, N_s, z_s^x)$ and verifies θ_s . If θ_s is not correct, C rejects S. Otherwise, C accepts k and decrypts ciphertext c'_s . C then checks if the identities C and S and nonces N_c and N_s in c'_s are decrypted correctly. If yes, C accepts the assigned IP address IP_C . Finally, C computes $\theta_c = H_1(C, S, N_c, N_s, \text{IP}_C, k, z_c, z_s, 11)$ and sends θ_c to S.

Round 6: S checks if θ_c is valid. If θ_c is not valid, S rejects C. Otherwise, S confirms that C has received the correct session key k and IP address IP_C . Next, S sends $\{k, \text{MAC}_C, \text{IP}_C\}$ to AP using a

secure channel.

10.2.3 Security Analysis of proposed authentication and key exchange protocol

In this section, we first prove that the proposed authentication and key exchange protocol is secure against dictionary attacks. Next, we show that an adversary is not able to obtain the session key k in the proposed authentication and key exchange protocol. We then prove that the adversary cannot obtain the client's identity. Lastly, we show that the proposed authentication and key exchange protocol is secure against DoS attack.

Theorem 38 *The proposed authentication and key exchange protocol is secure against off-line and on-line password guessing attack if the public-key encryption scheme PKE and message authentication code scheme MACS are secure.*

Proof: First, we consider the adversary $\mathcal{A}$ launching off-line password guessing attack on the authentication server S , then $\mathcal{A}$ must have been able to make a correct guess of the password pw using ciphertext $c_s = E_{p\kappa}(C, \text{MAC}_C, \text{pw}, S, N_c, N'_s\langle 1, b \rangle, N_s\langle b+1, L \rangle, \mu_c)$ in Round 3. As random nonce N_c , which is chosen by the client C , is unknown to $\mathcal{A}$ and furthermore, $\mathcal{A}$ does not know the private key $s\kappa$ of S to decrypt ciphertext c_s and the public-key encryption scheme PKE is assumed to be secure, therefore $\mathcal{A}$ is not able to verify the correct guess of the password pw using ciphertext c_s . Hence $\mathcal{A}$ is not able to launch off-line password guessing attack on the proposed protocol.

Next we consider the adversary $\mathcal{A}$ launching on-line password guessing attack on the authentication server S . Then in Round 3, $\mathcal{A}$ computes $N'_s\langle 1, b \rangle$ which satisfies $\lambda = H_1(S, N'_s\langle 1, b \rangle, N_s\langle b+1, L \rangle)$. Next, $\mathcal{A}$ selects random nonce N_c and $x \in Z_q^*$ and computes $z_c = g^x \bmod p$. Then $\mathcal{A}$ has to compute the valid $\mu_c = \mathcal{M}_\kappa(z_c, N_c, N_s, C, S, 00)$ and ciphertext $c'_s = E_{p\kappa}(C, \text{MAC}_C, \text{pw}, S, N_c, N'_s\langle 1, b \rangle, N_s\langle b+1, L \rangle, \mu'_c)$. As $\mathcal{A}$ does not know the secret key κ and the message authentication code scheme MACS is secure, $\mathcal{A}$ is not able to produce the valid μ_c . Therefore, S always reject μ_c . Hence the adversary $\mathcal{A}$ is not able to launch on-line password guess attacks on the proposed protocol. ■

Theorem 39 *An adversary $\mathcal{A}$ is not able to obtain the session key k in the proposed authentication and key exchange protocol if the message authentication code scheme MACS, cryptographic hash*

function H_1 and H_2 , symmetric key encryption scheme SKE and public-key encryption scheme PKE are secure and the CDH problem is difficult.

Proof: In this proof, we consider two cases (a) the adversary $\mathcal{A}$ tries to impersonate client C or server S to establish the session key k and (b) the adversary $\mathcal{A}$ tries to obtain k from the public information sent. For case (a), we first consider if $\mathcal{A}$ tries to impersonate client C. Then in Round 1, $\mathcal{A}$ sends HELLO message to S. In Round 3, $\mathcal{A}$ first computes $N'_s \langle 1, b \rangle$ which satisfies $\lambda = H_1(S, N'_s \langle 1, b \rangle, N_s \langle b + 1, L \rangle)$. $\mathcal{A}$ then selects random nonce N_c and $x \in Z_q^*$ and computes $z_c = g^x \bmod p$. Next, $\mathcal{A}$ needs to compute the valid $\mu_c = \mathcal{M}_\kappa(z_c, N_c, N_s, C, S, 00)$ and valid $c_s = E_{p\kappa}(C, \text{MAC}_C, \text{pw}, S, N_c, N'_s \langle 1, b \rangle, N_s \langle b + 1, L \rangle, \mu_c)$ using valid password pw . The server S also verifies the validity of μ_c using secret key κ . If μ_c is not valid, then S rejects C. As $\mathcal{A}$ does not know the password pw and by Theorem 38, it is shown that $\mathcal{A}$ is not able to obtain password pw from the proposed protocol through password guessing, therefore $\mathcal{A}$ is not able to compute the correct c_s using the valid password pw . Furthermore, $\mathcal{A}$ does not know the secret key κ and the message authentication code scheme MACS is assumed to be secure, therefore $\mathcal{A}$ is also not able to produce the valid μ_c . Hence $\mathcal{A}$ is not able to impersonate client C to establish the session key k . Next we consider if $\mathcal{A}$ is able to impersonate server S. In Round 2, $\mathcal{A}$ sends $\{S, N_s \langle b + 1, L \rangle, \lambda, b\}$ to C. Then in Round 4, $\mathcal{A}$ first computes $z_s = g^y \bmod p$. As $\mathcal{A}$ does not know the private key $s\kappa$ of S and the public-key encryption scheme PKE is assumed to be secure, hence $\mathcal{A}$ is not able to obtain N_c from ciphertext $c_s = E_{p\kappa}(C, \text{MAC}_C, \text{pw}, S, N_c, N'_s \langle 1, b \rangle, N_s \langle b + 1, L \rangle, \mu_c)$. Hence $\mathcal{A}$ selects a random N'_c and computes session key $k = H_2(C, S, N'_c, N_s, z_c^y)$, $\theta_s = H_1(S, C, N_s, N'_c, k, z_s, z_c, 01)$ and $c'_s = \mathcal{E}_k(C, S, \text{IP}_C, N'_c, N_s)$. Then $\mathcal{A}$ has to compute valid $\mu_s = \mathcal{M}_\kappa(z_s, \theta_s, N_s, N'_c, S, C, 10)$. The validity of μ_s is checked by C in Round 5. If μ_s is not valid, then C rejects S. As $\mathcal{A}$ does not know the secret key κ and the message authentication code scheme MACS is assumed to be secure, therefore $\mathcal{A}$ is not able to compute the valid μ_s . Hence $\mathcal{A}$ is not able to impersonate server S to establish the session key k . Therefore, $\mathcal{A}$ is not able to obtain session key k in case (a).

Now we consider case (b) as follows: Suppose the adversary $\mathcal{A}$ tries to obtain session key k from the public information, that is, z_s , z_c and N_s , sent in the proposed protocol. As random nonce N_c which is chosen by client C is unknown to $\mathcal{A}$ and furthermore, $\mathcal{A}$ is not able to compute g^{xy} from $z_s = g^y$ and $z_c = g^x$ as this is the CDH problem and is assumed to be difficult, therefore $\mathcal{A}$ is not

able to compute the session key $k = H_2(C||S||N_c||N_s||g^{xy})$. The adversary $\mathcal{A}$ is also not able to obtain session key k in Round 6 since k is sent to AP via a secure channel. ■

Lemma 4 *The adversary $\mathcal{A}$ is not able to obtain the identity of the client in the proposed authentication and key exchange protocol provided the public key encryption scheme PKE and symmetric key encryption scheme SKE are secure.*

Proof: First, consider the adversary $\mathcal{A}$ tries to obtain the client's identity from the ciphertext c_s in Round 3. As $\mathcal{A}$ does not know the private key $s\kappa$ of the server to decrypt c_s and the public key encryption scheme PKE is assumed to be secure, hence $\mathcal{A}$ is not able to obtain client's identity from the ciphertext c_s . Next, consider $\mathcal{A}$ tries to obtain the client's identity from c'_s in Round 4. As $\mathcal{A}$ does not know session key k used to encrypt/decrypt ciphertext c'_s and the symmetric key encryption scheme SKE is assumed to be secure, hence $\mathcal{A}$ is not able to obtain the client's identity from c'_s . The adversary is also unable to obtain client's identity from Round 6, which is sent to AP through a secure channel. ■

Theorem 40 *Assume the computation of hash function H_1 and message authentication code scheme MACS is efficient, then the proposed authentication and key exchange protocol is resistant to DoS attacks.*

Proof: First, consider the adversary launching DoS attacks by continuously sending HELLO message in Round 1. Then in Round 2, S only has to perform a cryptographic hash computation. Besides, if the adversary does not respond in Round 3, S will ignore the adversary after the expiry time has past. Hence the adversary will not be able to succeed in the DoS attack. Next consider the adversary that launches DoS attacks by continuously sending messages to S in Round 3. Then, S will detect the incorrect random nonce N_s , which the legitimate client has to compute using brute force, by checking if $N_s\langle 1, b \rangle = N'_s\langle 1, b \rangle$. The probability of the adversary making a correct guess of $N_s\langle 1, b \rangle$ is $\frac{1}{2^b}$. Otherwise, the adversary needs to make a brute force computation on $N_s\langle 1, b \rangle$. Furthermore, the adversary is not able to compute the correct μ_c as the adversary does not know the key κ . S would be able to detect the incorrect μ_c after decrypting ciphertext c_s . Hence the adversary cannot succeed in the DoS attack in Round 3. Finally, if the adversary launches a DoS attack in Round 5, then S can verify θ_c efficiently with just a hash operation. ■

Table 10.1: Message Size in bits

Round	1	2	3	4	5	6	Total
Wan et al's Protocol	1024	2240	2608	2240	336	-	8448
Proposed Protocol	32	372	1904	1920	160	336	4724

10.2.4 Complexity Analysis of proposed authentication and key exchange protocol

We compare the computational and communication costs of our proposed authentication and key exchange protocol with Wan et al's authentication and key exchange protocol [WDBA05]. For computation costs, there are no differences. The authentication server S performs 2 exponentiations and 1 public key decryption while client C performs 2 exponentiations and 1 public key encryption in both protocols. For communication costs, Table 10.1 shows the message sizes (in bits) in each round of both protocols. The client's MAC address MAC_C is 48-bits, IP address IP_C is 128-bits (IPv6) and we assume that the size of password pw is 64-bits. The length L of random N_s is 160-bits (i.e. $L = 160$ bits) and we also assume that $b = 20$ bits in Table 10.1. b is selected to be 20-bits so that the client would have to spend more time to compute $N_s \langle 1, b \rangle$ by brute force as compared to the computation time of a public key decryption operation by the server S in Round 3. From Table 10.1, it can be seen that all messages sent and received in every round of our proposed protocol are much smaller in size even though our protocol requires one more round. From the last column in Table 10.1, it can be seen that the total number of bits sent and received in Wan et al's protocol (8448 bits) is almost double of our proposed protocol (4724 bits).

10.3 Denial-of-Service Resilience Password-based Group Key Agreement Protocols for Wireless Networks

In the second part of this chapter, we present the proposed Password-based GKA protocols (Initial GKA (IKA) Protocol, Join Protocol and Leave Protocol) for the Centralized-Centralized (Central-Central) group model (described in Section 2.12.4 of Chapter 2) for an asymmetric wireless networks (e.g. Wi-Fi and WMNs) with n nodes. Without loss of generality, we assume that there are only

two layers in the Central-Central group model. We refer readers to Section 2.12.4 in Chapter 2 for more details of the Central-Central Group Model.

Let H_1 and H_2 be two cryptographic hash functions. We require a symmetric key encryption scheme SKE where $c = \mathcal{E}_k(m)$ denotes the ciphertext c of message m encrypted with key k . We also require a message authentication code scheme MACS where $\mu = \text{MAC}_k(m)$ denotes the authentication tag μ of message m using key k for the proposed protocols. Next, two random large primes, q and p , are selected such that q divides $p - 1$. (p is 1024 bits and q is 160 bits.) Two elements $g, h \in Z_p^*$ of order q are selected.

Initial Group Key Agreement (IKA)

We assume that during the setup phase prior to the group key agreement (group formation), each subgroup member $U_{i,j}$ for $1 \leq i \leq n_c$ and $1 \leq j \leq n_i$ holds a password $\text{pw}_{i,j}$ that it shares with subgroup controller S_i and is used for authentication between $U_{i,j}$ and S_i . Similarly, each subgroup controller S_i holds a password pw_i that it shares with main controller C and is used for authentication between S_i and C . The passwords $\text{pw}_{i,j}$ and pw_i are of low entropy and drawn uniformly at random from the dictionary `Password` of size $\hat{N}$ ($\hat{N}$ is often assumed to be small, typically less than a million). In addition, we assume that each $U_{i,j}$ and its subgroup controller S_i shares a pairwise secret key $\kappa_{i,j}$ that is securely pre-loaded into the tamper-proof device in $U_{i,j}$ and S_i during the setup phase. Similarly, we also assume that each subgroup controller S_i and main controller C shares a pairwise secret key κ_i that is securely pre-loaded into the tamper-proof device in S_i and C during the setup phase. The secret keys $\kappa_{i,j}$ and κ_i respectively are used as master keys to derive the authentication keys, $ak_{i,j} = H_2(\kappa_{i,j} || U_{i,j} || S_i || 01)$ and $ak_i = H_2(\kappa_i || S_i || C || 01)$ respectively, which are used to compute the message authentication codes and the encryption keys, $ek_{i,j} = H_2(\kappa_{i,j} || U_{i,j} || S_i || 10)$ and $ek_i = H_2(\kappa_i || S_i || C || 10)$ respectively, which are used for symmetric key encryption in the GKA. Each $U_{i,j}$ and S_i will store $ak_{i,j}$ and $ek_{i,j}$ in the tamper-proof device and each S_i and C will also store ak_i and ek_i in the tamper-proof device. Each $U_{i,j}$ will keep password $\text{pw}_{i,j}$ in mind while S_i will store passwords $\text{pw}_{i,j}$ for $1 \leq j \leq n_i$ in a tamper-proof device. Similarly, each S_i and C will store passwords pw_i for $1 \leq i \leq n_c$ in a tamper-proof device. As each $U_{i,j}$ keeps his password $\text{pw}_{i,j}$ in mind, even if the adversary captures the user device with the secret key, the

adversary is still not able to forge user $U_{i,j}$ as the adversary does not know the password $\text{pw}_{i,j}$.

The protocol consists of 3 phases. In Phase 1, the subgroup controllers S_i will interact with its subgroup members $U_{i,j}$ and compute the subgroup key K_i . Next in Phase 2, S_i will use subgroup key K_i to interact with main controller C to obtain the final group key K . Finally in Phase 3, the final group key K will be sent down securely by C to the various S_i who will in turn send its subgroup key K_i and final group key K securely to its subgroup members $U_{i,j}$. Key confirmation messages are also sent together to verify and confirm the subgroup key K_i and final group key K . The details of the proposed Password-based GKA protocol is described as follows:

Phase 1: Subgroup Key Generation

Round 1: Each subgroup controller S_i for $1 \leq i \leq n_c$ randomly selects $\tau_i \in Z_q^*$, $\tau_{i,j} \in Z_q^*$ for $1 \leq j \leq n_i$ and session identity SID_i , encrypts $\tau_i || \tau_{i,j} || S_i || U_{i,j} || \text{SID}_i$ using $ek_{i,j}$ and obtains ciphertext $c_{i,j} = \mathcal{E}_{ek_{i,j}}(\tau_i || \tau_{i,j} || S_i || U_{i,j} || \text{SID}_i)$ and computes $v_{i,j} = H_1(\text{pw}_{i,j} || \tau_i || \tau_{i,j} || \text{SID}_i)$ for $1 \leq j \leq n_i$. Finally, S_i broadcasts $\{\text{SID}_i || S_i || c_{i,j} || v_{i,j}\}$ for $1 \leq j \leq n_i$ to subgroup $\mathcal{G}_i$.

Round 2: Each subgroup member $U_{i,j}$ for $1 \leq i \leq n_c$ and $1 \leq j \leq n_i$ first decrypts $c_{i,j}$ and checks if the identities S_i , $U_{i,j}$ and SID_i are decrypted correctly. If the identities are not correct, $U_{i,j}$ rejects S_i . Otherwise, $U_{i,j}$ uses password $\text{pw}_{i,j}$ to check the validity of $v_{i,j} = H_1(\text{pw}_{i,j} || \tau_i || \tau_{i,j} || \text{SID}_i)$. If it is incorrect, $U_{i,j}$ rejects S_i . Otherwise, $U_{i,j}$ randomly selects $r_{i,j} \in Z_q^*$ and computes

$$z_{i,j} = g^{r_{i,j}} \bmod p, \quad z_{i,j}^* = z_{i,j} \cdot h^{H_1(\text{SID}_i || U_{i,j} || S_i || \text{pw}_{i,j} || \tau_i || \tau_{i,j})}$$

$$\mu_{i,j} = \text{MAC}_{ak_{i,j}}(\tau_i || \tau_{i,j} || z_{i,j}^* || \text{SID}_i || U_{i,j} || S_i || 00)$$

$$\bar{\mu}_{i,j} = \text{MAC}_{ak_{i,j}}(\tau_i || \tau_{i,j} || z_{i,j} || \text{SID}_i || U_{i,j} || S_i || 01)$$

Finally, $U_{i,j}$ sends $\{\text{SID}_i || U_{i,j} || z_{i,j}^* || \mu_{i,j} || \bar{\mu}_{i,j}\}$ to S_i .

Round 3: Each S_i for $1 \leq i \leq n_c$ first checks if $\mu_{i,j} = \text{MAC}_{ak_{i,j}}(\tau_i || \tau_{i,j} || z_{i,j}^* || \text{SID}_i || U_{i,j} || S_i || 00)$ for $1 \leq j \leq n_i$. If the check is not correct, S_i blocks malicious user $U_{i,j}$. Otherwise, S_i obtains $z_{i,j} = z_{i,j}^* \cdot h^{-H_1(\text{SID}_i || U_{i,j} || S_i || \text{pw}_{i,j} || \tau_i || \tau_{i,j})}$ and checks $\bar{\mu}_{i,j} = \text{MAC}_{ak_{i,j}}(\tau_i || \tau_{i,j} || z_{i,j} || \text{SID}_i || U_{i,j} || S_i || 01)$. If the check is incorrect, S_i blocks malicious user $U_{i,j}$. Otherwise, S_i selects random $r_i \in Z_q^*$ and computes $z_i = g^{r_i} \bmod p$. Next, S_i computes the Diffie-Hellman (DH) keys $t_{i,j}$ for $1 \leq j \leq n_i$ that

it shares with $U_{i,j}$ and subgroup key K_i :

$$\begin{aligned} t_{i,j} &= z_{i,j}^{r_i} \bmod p \\ K_i &= \left(z_i \cdot \prod_{j=1}^{n_i} z_{i,j} \right)^{r_i} \bmod p \end{aligned}$$

Phase 2: Final Key Generation

Round 1: Main controller C randomly selects $\tau_C \in Z_q^*$, $\bar{\tau}_i \in Z_q^*$ for $1 \leq i \leq n_c$ and session identity SID_c , encrypts $\tau_C || \bar{\tau}_i || C || S_i || SID_c$ using ek_i to obtain $c_i = \mathcal{E}_{ek_i}(\tau_C || \bar{\tau}_i || C || S_i || SID_c)$ and computes $v_i = H_1(pw_i || \tau_C || \bar{\tau}_i || SID_c)$ for $1 \leq i \leq n_c$. Finally, C broadcasts $\{SID_c || C || c_i || v_i\}$ for $1 \leq i \leq n_c$ to subgroup controllers S_i .

Round 2: Each subgroup controller S_i for $1 \leq i \leq n_c$ first decrypts c_i and checks if the identities C , S_i and SID_c are decrypted correctly. If the identities are not correct, S_i rejects C . Otherwise, S_i uses password pw_i to check if $v_i = H_1(pw_i || \tau_C || \bar{\tau}_i || SID_c)$. If the check is incorrect, S_i rejects C . Otherwise, S_i uses the subgroup key K_i computed in Phase 1 Round 3 to compute:

$$\begin{aligned} y_i &= g^{H_2(K_i)} \bmod p, \quad y_i^* = y_i \cdot h^{H_1(SID_c || S_i || C || pw_i || \tau_C || \bar{\tau}_i)} \\ \mu_i &= MAC_{ak_i}(\tau_C || \bar{\tau}_i || y_i^* || SID_c || S_i || C || 00) \\ \bar{\mu}_i &= MAC_{ak_i}(\tau_C || \bar{\tau}_i || y_i || SID_c || S_i || C || 01) \end{aligned}$$

Finally, S_i sends $\{SID_c || S_i || y_i^* || \mu_i || \bar{\mu}_i\}$ to C .

Round 3: Main controller C first checks if $\mu_i = MAC_{ak_i}(\tau_C || \bar{\tau}_i || y_i^* || SID_c || S_i || C || 00)$ for $1 \leq i \leq n_c$. If the check is not correct, C blocks malicious subgroup controller S_i . Otherwise, C obtains $y_i = y_i^* \cdot h^{-H_1(SID_c || S_i || C || pw_i || \tau_C || \bar{\tau}_i)}$ and checks $\bar{\mu}_i = MAC_{ak_i}(\tau_C || \bar{\tau}_i || y_i || SID_c || S_i || C || 01)$. If the check is incorrect, C blocks malicious subgroup controller S_i . Otherwise, C selects random $r_C \in Z_q^*$ and computes $y_C = g^{r_C} \bmod p$. Next, C computes the DH keys t_i that it shares with each S_i , the final

group key K , key confirmation code θ_C and $\hat{\mu}_i$:

$$\begin{aligned} t_i &= y_i^{r_C} \bmod p \\ K &= \left(y_C \cdot \prod_{i=1}^{n_C} y_i \right)^{r_C} \bmod p \\ \theta_C &= H_1(\text{SID}_C || \text{UID}_C || K || \tau_C || y_C) \\ \hat{\mu}_i &= \text{MAC}_{ak_i}(\tau_C || \bar{\tau}_i || y_C || \theta_C || \text{SID}_C || C || S_i || 11) \end{aligned}$$

where $\text{UID}_C = C || S_1 || S_2 || \dots || S_{n_C}$. C encrypts $K || C || y_i || \text{SID}_C$ using t_i and obtains $\hat{c}_i = \mathcal{E}_{t_i}(K || C || y_i || \text{SID}_C)$ for $1 \leq i \leq n_C$. Finally, C broadcast $\{\text{SID}_C || \text{UID}_C || C || y_C || \hat{\mu}_i || \hat{c}_i || \theta_C\}$ for $1 \leq i \leq n_C$.

Phase 3: Key Distribution

Round 1: Each S_i for $1 \leq i \leq n_C$ first checks if $\hat{\mu}_i = \text{MAC}_{ak_i}(\tau_C || \bar{\tau}_i || y_C || \theta_C || \text{SID}_C || C || S_i || 11)$ is correct. If the equation is incorrect, S_i rejects malicious main controller C . Otherwise, S_i computes $t_i = y_C^{H_2(K_i)} \bmod p$ and uses t_i to decrypt $\hat{c}_i$ and obtains the final group key K , identity C , y_i and SID_C . If the identity C , y_i and SID_C are not decrypted correctly, then S_i rejects $\hat{c}_i$. Otherwise, S_i confirms and accepts the final group key K if the key confirmation code $\theta_C = H_1(\text{SID}_C || \text{UID}_C || K || \tau_C || y_C)$ holds. Next, S_i encrypts $K_i || K || S_i || z_{i,j} || \text{SID}_i$ using $t_{i,j}$ and obtains $\hat{c}_{i,j} = \mathcal{E}_{t_{i,j}}(K_i || K || S_i || z_{i,j} || \text{SID}_i)$ for $1 \leq j \leq n_i$. S_i then computes the key confirmation code θ_i and $\hat{\mu}_{i,j}$ for $1 \leq j \leq n_i$:

$$\begin{aligned} \theta_i &= H_1(\text{SID}_i || \text{UID}_i || K_i || K || \tau_i || z_i) \\ \hat{\mu}_{i,j} &= \text{MAC}_{ak_{i,j}}(\tau_i || \tau_{i,j} || z_i || \theta_i || \text{SID}_i || S_i || U_{i,j} || 11) \end{aligned}$$

where $\text{UID}_i = S_i || U_{i,1} || U_{i,2} || \dots || U_{i,n_i}$. Finally, S_i broadcast $\{\text{SID}_i || \text{UID}_i || S_i || z_i || \hat{\mu}_{i,j} || \hat{c}_{i,j} || \theta_i\}$ for $1 \leq j \leq n_i$ to subgroup $\mathcal{G}_i$.

Round 2: Each $U_{i,j}$ for $1 \leq i \leq n_C$ and $1 \leq j \leq n_i$ checks if $\hat{\mu}_{i,j} = \text{MAC}_{ak_{i,j}}(\tau_i || \tau_{i,j} || z_i || \theta_i || \text{SID}_i || S_i || U_{i,j} || 11)$ is valid. If the equation is invalid, $U_{i,j}$ rejects malicious subgroup controller S_i . Otherwise, $U_{i,j}$ computes $t_{i,j} = z_i^{r_{i,j}} \bmod p$ and uses $t_{i,j}$ to decrypt $\hat{c}_{i,j}$ and obtains subgroup key K_i , final group key K , identity S_i , $z_{i,j}$ and SID_i . If the identity S_i , $z_{i,j}$ and SID_i are not decrypted correctly, $U_{i,j}$ rejects $\hat{c}_{i,j}$. Otherwise, $U_{i,j}$ confirms and accepts both subgroup key K_i and final group key K if the key confirmation code $\theta_i = H_1(\text{SID}_i || \text{UID}_i || K_i || K || \tau_i || z_i)$ holds.

Join Protocol

Suppose a set of m new users $\mathcal{J} = \{\mathbf{U}_{n+1}, \dots, \mathbf{U}_{n+m}\}$ wishes to join subgroup $\mathcal{G}_J = \{\mathbf{U}_{J,1}, \dots, \mathbf{U}_{J,n_J}\}$. The Join Protocol will be executed to establish a new subgroup key K'_J for subgroup $\mathcal{G}'_J = \mathcal{G}_J \cup \mathcal{J} = \{\mathbf{U}_{J,1}, \dots, \mathbf{U}_{J,n_J+m}\}$, where $\mathbf{U}_{J,j}$ for $1 \leq j \leq n_J$ are the existing users and $\mathbf{U}_{J,k}$ for $n_J+1 \leq k \leq n_J+m$ are the new users, and new final group key K' for all users in the network to prevent the new users from knowing previous group communications. We assume that each new user $\mathbf{U}_{J,k}$ for $n_J+1 \leq k \leq n_J+m$ shares password $\mathbf{pw}_{J,k}$ with subgroup controller $\mathbf{S}_J$. We also assume that $\mathbf{U}_{J,k}$ and $\mathbf{S}_J$ shares pairwise secret key $\kappa_{J,k}$, which is used to derive authentication key $ak_{J,k} = \mathbf{H}_2(\kappa_{J,k} || \mathbf{U}_{J,k} || \mathbf{S}_J || 01)$ and the encryption key $ek_{J,k} = \mathbf{H}_2(\kappa_{J,k} || \mathbf{U}_{J,k} || \mathbf{S}_J || 10)$. Each $\mathbf{U}_{J,k}$ and $\mathbf{S}_J$ stores $\kappa_{J,k}$, $ak_{J,k}$ and $ek_{J,k}$ securely in tamper-proof devices.

The Join Protocol consists of 3 phases. During Phase 1, subgroup controller $\mathbf{S}_J$ of subgroup $\mathcal{G}'_J$ and new users $\mathbf{U}_{J,k}$ for $n_J+1 \leq k \leq n_J+m$ will interact and compute the new subgroup key K'_J . Next in Phase 2, $\mathbf{S}_J$ uses K'_J to interact with main controller $\mathbf{C}$ in order for $\mathbf{C}$ to compute the new final group key K' . Finally in Phase 3, the new final group key K' is disseminated from $\mathbf{C}$ to all user nodes via subgroup controllers $\mathbf{S}_i$ for $1 \leq i \leq n_c$ (including $\mathbf{S}_J$). $\mathbf{S}_J$ will then send the new subgroup key K'_J and new final group key K' securely to all existing users $\mathbf{U}_{J,j}$ for $1 \leq j \leq n_J$ and new users $\mathbf{U}_{J,k}$ for $n_J+1 \leq k \leq n_J+m$. $\mathbf{S}_i$ for $1 \leq i \neq J \leq n_c$ will also update its subgroup with the new final group key K' . Key confirmation messages are also sent together to verify and confirm the new subgroup key K'_J and the new final group key K' . The Join Protocol is described below:

Phase 1: Subgroup Key Generation

Round 1: Subgroup controller $\mathbf{S}_J$ randomly selects new $\tau'_J \in Z_q^*$, $\tau'_{J,k} \in Z_q^*$ for $n_J+1 \leq k \leq n_J+m$ and session identity SID'_J , encrypts $\tau'_J || \tau'_{J,k} || \mathbf{S}_J || \mathbf{U}_{J,k} || \text{SID}'_J$ using $ek_{J,k}$ and obtains ciphertext $\mathbf{c}'_{J,k} = \mathcal{E}_{ek_{J,k}}(\tau'_J || \tau'_{J,k} || \mathbf{S}_J || \mathbf{U}_{J,k} || \text{SID}'_J)$. $\mathbf{S}_J$ then computes $v'_{J,k} = \mathbf{H}_1(\mathbf{pw}_{J,k} || \tau'_J || \tau'_{J,k} || \text{SID}'_J)$ and broadcasts $\{\text{SID}'_J || \mathbf{S}_J || \mathbf{c}'_{J,k} || v'_{J,k}\}$ for $n_J+1 \leq k \leq n_J+m$ to the new users $\mathbf{U}_{J,k}$ for $n_J+1 \leq k \leq n_J+m$.

Round 2: Each new user $\mathbf{U}_{J,k}$ for $n_J+1 \leq k \leq n_J+m$ first decrypts $\mathbf{c}'_{J,k}$ and checks if the identities $\mathbf{S}_J$, $\mathbf{U}_{J,k}$ and SID'_J are decrypted correctly. If the identities are not correct, $\mathbf{U}_{J,k}$ rejects $\mathbf{S}_J$. Otherwise, $\mathbf{U}_{J,k}$ uses password $\mathbf{pw}_{J,k}$ to check the validity of $v'_{J,k} = \mathbf{H}_1(\mathbf{pw}_{J,k} || \tau'_J || \tau'_{J,k} || \text{SID}'_J)$. If

it is incorrect, $U_{J,k}$ rejects S_J . Otherwise, $U_{J,k}$ randomly selects $r'_{J,k} \in Z_q^*$ and computes

$$\begin{aligned} z'_{J,k} &= g^{r'_{J,k}} \bmod p, \quad (z^*_{J,k})' = z'_{J,k} \cdot h^{H_1(\text{SID}'_J || U_{J,k} || S_J || \text{pw}_{J,k} || \tau'_J || \tau'_{J,k})} \\ \mu'_{J,k} &= \text{MAC}_{ak_{J,k}}(\tau'_J || \tau'_{J,k} || (z^*_{J,k})' || \text{SID}'_J || U_{J,k} || S_J || 00) \\ \bar{\mu}'_{J,k} &= \text{MAC}_{ak_{J,k}}(\tau'_J || \tau'_{J,k} || z'_{J,k} || \text{SID}'_J || U_{J,k} || S_J || 01) \end{aligned}$$

Finally, $U_{J,k}$ sends $\{\text{SID}'_J || U_{J,k} || (z^*_{J,k})' || \mu'_{J,k} || \bar{\mu}'_{J,k}\}$ to S_J .

Round 3: S_J checks if $\mu'_{J,k} = \text{MAC}_{ak_{J,k}}(\tau'_J || \tau'_{J,k} || (z^*_{J,k})' || \text{SID}'_J || U_{J,k} || S_J || 00)$. If the check is not correct, S_J blocks malicious user $U_{J,k}$. Otherwise, S_J obtains $z'_{J,k} = (z^*_{J,k})' \cdot h^{-H_1(\text{SID}'_J || U_{J,k} || S_J || \text{pw}_{J,k} || \tau'_J || \tau'_{J,k})}$ and checks $\bar{\mu}'_{J,k} = \text{MAC}_{ak_{J,k}}(\tau'_J || \tau'_{J,k} || z'_{J,k} || \text{SID}'_J || U_{J,k} || S_J || 01)$. If the check is incorrect, S_J blocks malicious user $U_{J,k}$. Otherwise, S_J selects new random $r'_J \in Z_q^*$ and computes $z'_J = g^{r'_J} \bmod p$. Next, S_J computes DH keys $t'_{J,k}$ for $n_J + 1 \leq k \leq n_J + m$ shared with $U_{J,k}$ and new subgroup key K'_J :

$$\begin{aligned} t'_{J,k} &= (z'_{J,k})^{r'_J} \bmod p \\ K'_J &= \left(z'_J \cdot \prod_{j=1}^{n_J} z_{J,j} \cdot \prod_{k=n_J+1}^{n_J+m} z'_{J,k} \right)^{r'_J} \bmod p \end{aligned}$$

Phase 2: Final Key Generation

Round 1: Main controller C selects random $\tau'_C \in Z_q^*$, $\bar{\tau}'_J \in Z_q^*$, $\bar{\tau}'_i \in Z_q^*$ for $1 \leq i \neq J \leq n_c$ and session identity SID'_C , encrypts $\tau'_C || \bar{\tau}'_J || C || S_J || \text{SID}'_C$ using ek_J and $\tau'_C || \bar{\tau}'_i || C || S_i || \text{SID}'_C$ using ek_i to obtain $c'_J = \mathcal{E}_{ek_J}(\tau'_C || \bar{\tau}'_J || C || S_J || \text{SID}'_C)$ and $c'_i = \mathcal{E}_{ek_i}(\tau'_C || \bar{\tau}'_i || C || S_i || \text{SID}'_C)$ for $1 \leq i \neq J \leq n_c$ respectively. Next, C computes $v'_J = H_1(\text{pw}_J || \tau'_C || \bar{\tau}'_J || \text{SID}'_C)$ and $v'_i = H_1(\text{pw}_i || \tau'_C || \bar{\tau}'_i || \text{SID}'_C)$ for $1 \leq i \neq J \leq n_c$ and sends $\{\text{SID}'_C || C || c'_J || v'_J || c'_i || v'_i\}$ for $1 \leq i \neq J \leq n_c$ to subgroup controllers S_J and S_i for $1 \leq i \neq J \leq n_c$.

Round 2:

(i) The subgroup controller S_J first decrypts c'_J and checks if the identities C , S_J and SID'_C are decrypted correctly. If the identities are not correct, S_J rejects C . Otherwise, S_J uses password pw_J to check if $v'_J = H_1(\text{pw}_J || \tau'_C || \bar{\tau}'_J || \text{SID}'_C)$ is correct. If the check is incorrect, S_J rejects C . Otherwise,

S_J uses new subgroup key K'_J computed in Phase 1 Round 3 to compute:

$$\begin{aligned} y'_J &= g^{\text{H}_2(K'_J)} \bmod p, \quad (y_J^*)' = y'_J \cdot h^{\text{H}_1(\text{SID}'_c || S_J || C || \text{pw}_J || \tau'_c || \bar{\tau}'_J)} \\ \mu'_J &= \text{MAC}_{ak_J}(\tau'_c || \bar{\tau}'_J || (y_J^*)' || \text{SID}'_c || S_J || C || 00) \\ \bar{\mu}'_J &= \text{MAC}_{ak_J}(\tau'_c || \bar{\tau}'_J || y'_J || \text{SID}'_c || S_J || C || 01) \end{aligned}$$

Finally, S_J sends $\{\text{SID}'_c || S_J || (y_J^*)' || \mu'_J || \bar{\mu}'_J\}$ to C .

(ii) Subgroup controller S_i for $1 \leq i \neq J \leq n_c$ first decrypts c'_i and checks if the identities C , S_i and SID'_c are decrypted correctly. If the identities are not correct, S_i rejects C . Otherwise, S_i uses password pw_i to check if $v'_i = \text{H}_1(\text{pw}_i || \tau'_c || \bar{\tau}'_i || \text{SID}'_c)$ is correct. If the check is incorrect, S_i rejects C . Otherwise, S_i stores τ'_c , $\bar{\tau}'_i$ and SID'_c for later computation.

Round 3: Main controller C first checks if $\mu'_J = \text{MAC}_{ak_J}(\tau'_c || \bar{\tau}'_J || (y_J^*)' || \text{SID}'_c || S_J || C || 00)$. If the check is not correct, C blocks malicious subgroup controller S_J . Otherwise, C obtains $y'_J = (y_J^*)' \cdot h^{-\text{H}_1(\text{SID}'_c || S_J || C || \text{pw}_J || \tau'_c || \bar{\tau}'_J)}$ and checks $\bar{\mu}'_J = \text{MAC}_{ak_J}(\tau'_c || \bar{\tau}'_J || y'_J || \text{SID}'_c || S_J || C || 01)$. If the check is incorrect, C blocks malicious subgroup controller S_J . Otherwise, C selects new random $r'_c \in Z_q^*$ and computes $y'_c = g^{r'_c} \bmod p$. Next, C computes the new DH keys t'_J and t'_i that it shares with S_J and S_i for $1 \leq i \neq J \leq n_c$ respectively, the new final group key K' , θ'_c , $\hat{\mu}'_J$ and $\hat{\mu}'_i$ for $1 \leq i \neq J \leq n_c$:

$$\begin{aligned} t'_J &= (y'_J)^{r'_c} \bmod p, \quad t'_i = y_i^{r'_c} \bmod p \\ K' &= \left(y'_c \cdot y'_J \cdot \prod_{i=1, i \neq J}^{n_c} y_i \right)^{r'_c} \bmod p \\ \theta'_c &= \text{H}_1(\text{SID}'_c || \text{UID}_C || K' || \tau'_c || y'_c) \\ \hat{\mu}'_J &= \text{MAC}_{ak_J}(\tau'_c || \bar{\tau}'_J || y'_c || \theta'_c || \text{SID}'_c || C || S_J || 11) \\ \hat{\mu}'_i &= \text{MAC}_{ak_i}(\tau'_c || \bar{\tau}'_i || y'_c || \theta'_c || \text{SID}'_c || C || S_i || 11) \end{aligned}$$

where $\text{UID}_C = C || S_1 || S_2 || \dots || S_{n_c}$. C encrypts $K' || C || y'_J || \text{SID}'_c$ using t'_J and $K' || C || y_i || \text{SID}'_c$ using t'_i for $1 \leq i \neq J \leq n_c$ to obtain $\hat{c}'_J = \mathcal{E}_{t'_J}(K' || C || y'_J || \text{SID}'_c)$ and $\hat{c}'_i = \mathcal{E}_{t'_i}(K' || C || y_i || \text{SID}'_c)$ for $1 \leq i \neq J \leq n_c$ respectively. C then broadcast $\{\text{SID}'_c || \text{UID}_C || C || y'_c || \hat{\mu}'_J || \hat{\mu}'_i || \hat{c}'_J || \hat{c}'_i || \theta'_c\}$ for $1 \leq i \neq J \leq n_c$.

Phase 3: Key Distribution

Round 1:

(i) S_J first checks if $\hat{\mu}'_J = \text{MAC}_{ak_J}(\tau'_C || \bar{\tau}'_J || y'_C || \theta'_C || \text{SID}'_C || C || S_J || 11)$ is correct. If the equation is incorrect, then S_J rejects C . Otherwise, S_J computes $t'_J = (y'_C)^{H_2(K'_J)} \bmod p$ and uses t'_J to decrypt $\hat{c}'_J$ and obtains the new final group key K' , identity C , y'_J and SID'_C . If the identity C , y'_J and SID'_C are not decrypted correctly, S_J rejects $\hat{c}'_J$. Otherwise, S_J confirms and accepts the new final group key K' if the key confirmation code $\theta'_C = H_1(\text{SID}'_C || \text{UID}_C || K' || \tau'_C || y'_C)$ holds. Next, S_J encrypts $K'_J || K' || S_J || z'_{J,k} || \text{SID}'_J$ using $t'_{J,k}$ and $K'_J || K' || S_J || \text{SID}'_J || \tau'_J$ using existing subgroup key K_J and obtains $\hat{c}'_{J,k} = \mathcal{E}_{t'_{J,k}}(K'_J || K' || S_J || z'_{J,k} || \text{SID}'_J)$ for $n_J + 1 \leq k \leq n_J + m$ and $\hat{c}''_J = \mathcal{E}_{K_J}(K'_J || K' || S_J || \text{SID}'_J || \tau'_J)$. S_J then computes the key confirmation code θ'_J and $\hat{\mu}'_{J,k}$ for $n_J + 1 \leq k \leq n_J + m$:

$$\begin{aligned}\theta'_J &= H_1(\text{SID}'_J || \text{UID}'_J || K'_J || K' || \tau'_J || z'_J) \\ \hat{\mu}'_{J,k} &= \text{MAC}_{ak_{J,k}}(\tau'_J || \tau'_{J,k} || z'_J || \theta'_J || \text{SID}'_J || S_J || U_{J,k} || 11)\end{aligned}$$

where $\text{UID}'_J = S_J || U_{J,1} || U_{J,2} || \dots || U_{J,n_J+m}$. Next, S_J broadcast $\{\text{SID}'_J || \text{UID}'_J || S_J || z'_J || \hat{\mu}'_{J,k} || \hat{c}'_{J,k} || \hat{c}''_J || \theta'_J\}$ for $n_J + 1 \leq k \leq n_J + m$ to subgroup $\mathcal{G}'_J$.

(ii) Each S_i for $1 \leq i \neq J \leq n_c$ first checks if $\hat{\mu}'_i = \text{MAC}_{ak_i}(\tau'_C || \bar{\tau}'_i || y'_C || \theta'_C || \text{SID}'_C || C || S_i || 11)$ is correct. If the equation is incorrect, S_i rejects C . Otherwise, S_i computes $t'_i = (y'_C)^{H_2(K'_i)} \bmod p$ and uses t'_i to decrypt $\hat{c}'_i$ and obtains the new final group key K' , identity C , y_i and SID'_C . If the identity C , y_i and SID'_C are not decrypted correctly, S_i rejects $\hat{c}'_i$. Otherwise, S_i confirms and accepts the final group key K' if the key confirmation code $\theta'_C = H_1(\text{SID}'_C || \text{UID}_C || K' || \tau'_C || y'_C)$ holds. Next, S_i selects $\tau'_i \in Z_q^*$ and session identity SID'_i and encrypts $K' || S_i || \text{SID}'_i || \tau'_i$ using K_i and obtains $\hat{c}''_i = \mathcal{E}_{K_i}(K' || S_i || \text{SID}'_i || \tau'_i)$. S_i then computes the key confirmation code $\theta'_i = H_1(\text{SID}'_i || \text{UID}_i || K' || \tau'_i || z_i)$, where $\text{UID}_i = S_i || U_{i,1} || U_{i,2} || \dots || U_{i,n_i}$. Finally, S_i broadcast $\{\text{SID}'_i || \text{UID}_i || S_i || \hat{c}''_i || \theta'_i\}$ to subgroup $\mathcal{G}_i$.

Round 2:

(i) Each new user $U_{J,k}$ for $n_J + 1 \leq k \leq n_J + m$ checks if $\hat{\mu}'_{J,k} = \text{MAC}_{ak_{J,k}}(\tau'_J || \tau'_{J,k} || z'_J || \theta'_J || \text{SID}'_J || S_J || U_{J,k} || 11)$ is valid. If the equation is invalid, $U_{J,k}$ rejects the malicious subgroup controller S_J . Otherwise, $U_{J,k}$ computes $t'_{J,k} = (z'_J)^{r'_{J,k}} \bmod p$ and uses $t'_{J,k}$ to decrypt $\hat{c}'_{J,k}$ and obtains subgroup key K'_J , new final group key K' , identity S_J , $z'_{J,k}$ and SID'_J . If the identity S_J , $z'_{J,k}$ and SID'_J are not decrypted correctly, $U_{J,k}$ rejects $\hat{c}'_{J,k}$. Otherwise, $U_{J,k}$ confirms and accepts new subgroup key

K'_J and final group key K' if $\theta'_J = H_1(\text{SID}'_J || \text{UID}'_J || K'_J || K' || \tau'_J || z'_J)$ holds.

(ii) Each existing user $U_{J,j}$ for $1 \leq j \leq n_J$ uses existing subgroup key K_J to decrypt $\hat{c}''_J$ and obtains new subgroup key K'_J , new final group key K' , identity S_J , SID'_J and τ'_J . If the identity S_J and SID'_J are not decrypted correctly, $U_{J,j}$ rejects $\hat{c}''_J$. Otherwise, $U_{J,j}$ confirms and accepts new subgroup key K'_J and new final group key K' if $\theta'_J = H_1(\text{SID}'_J || \text{UID}'_J || K'_J || K' || \tau'_J || z'_J)$ holds.

(iii) Each $U_{i,j}$ for $1 \leq i \neq J \leq n_c$ and $1 \leq j \leq n_i$ uses K_i to decrypt $\hat{c}''_i$ and obtains new final group key K' , identity S_i , SID'_i and τ'_i . If the identity S_i and SID'_i are not decrypted correctly, $U_{i,j}$ rejects $\hat{c}''_i$. Otherwise, $U_{i,j}$ confirms and accepts new final group key K' if $\theta'_i = H_1(\text{SID}'_i || \text{UID}_i || K' || \tau'_i || z_i)$ holds.

Leave Protocol

Suppose a set of ℓ users $\mathcal{L} = \{U_{L,l_1}, \dots, U_{L,l_\ell}\}$, with index $\{l_1, \dots, l_\ell\} \subset \{1, 2, \dots, n_L\}$, wishes to leave subgroup $\mathcal{G}_L = \{U_{L,1}, \dots, U_{L,n_L}\}$, then the Leave Protocol will be executed to establish a new subgroup key K'_L and new final group key K' to protect future group communications from the leaving users. We re-index the remaining users in subgroup $\mathcal{G}'_L = \mathcal{G}_L \setminus \mathcal{L} = \{V_{L,1}, \dots, V_{L,n_L-\ell}\}$.

The Leave Protocol consists of 3 phases. During Phase 1, subgroup controller S_L computes the new subgroup key K'_L . Next in Phase 2, S_L uses K'_L to interact with main controller C in order for C to compute the new final group key K' . Finally in Phase 3, the new final group key K' is disseminated from C to all user nodes via subgroup controllers S_i for $1 \leq i \leq n_c$ (including S_L). S_L will send the new subgroup key K'_L and new final group key K' securely to remaining users $V_{L,j}$ for $1 \leq j \leq n_L - \ell$. S_i for $1 \leq i \neq L \leq n_c$ will also update its subgroup with the new final group key K' . Key confirmation messages are also sent together to verify and confirm the new subgroup key K'_L and new final group key K' . The Leave Protocol works as follows:

Phase 1: Subgroup Key Generation

Round 1: S_L selects new random $r'_L \in Z_q^*$ and computes $z'_L = g^{r'_L} \bmod p$. Next, S_L computes new

DH keys $t'_{L,j}$ shared with remaining users $\mathbf{V}_{L,j}$ for $1 \leq j \leq n_L - \ell$ and new subgroup key K'_L :

$$\begin{aligned} t'_{L,j} &= z'^{r'_L}_{L,j} \bmod p \\ K'_L &= \left(z'_L \cdot \prod_{j=1}^{n_L-\ell} z_{L,j} \right)^{r'_L} \bmod p \end{aligned}$$

where $z_{L,j}$ belongs to remaining users $\mathbf{V}_{L,j}$ for $1 \leq j \leq n_L - \ell$.

Phase 2: Final Key Generation

Round 1: Main controller $\mathbf{C}$ randomly selects $\tau'_C \in Z_q^*$, $\bar{\tau}'_L \in Z_q^*$, $\bar{\tau}'_i \in Z_q^*$ for $1 \leq i \neq L \leq n_c$ and session identity SID'_c , encrypts $\tau'_C || \bar{\tau}'_L || \mathbf{C} || \mathbf{S}_L || \text{SID}'_c$ using ek_L and $\tau'_C || \bar{\tau}'_i || \mathbf{C} || \mathbf{S}_i || \text{SID}'_c$ using ek_i and obtains $c'_L = \mathcal{E}_{ek_L}(\tau'_C || \bar{\tau}'_L || \mathbf{C} || \mathbf{S}_L || \text{SID}'_c)$ and $c'_i = \mathcal{E}_{ek_i}(\tau'_C || \bar{\tau}'_i || \mathbf{C} || \mathbf{S}_i || \text{SID}'_c)$ for $1 \leq i \neq L \leq n_c$ respectively. Next, $\mathbf{C}$ computes $v'_L = \mathbf{H}_1(\mathbf{pw}_L || \tau'_C || \bar{\tau}'_L || \text{SID}'_c)$ and $v'_i = \mathbf{H}_1(\mathbf{pw}_i || \tau'_C || \bar{\tau}'_i || \text{SID}'_c)$ for $1 \leq i \neq L \leq n_c$ and sends $\{\text{SID}'_c || \mathbf{C} || c'_L || v'_L || c'_i || v'_i\}$ for $1 \leq i \neq L \leq n_c$ to subgroup controllers $\mathbf{S}_L$ and $\mathbf{S}_i$ for $1 \leq i \neq L \leq n_c$.

Round 2:

(i) The subgroup controller $\mathbf{S}_L$ first decrypts c'_L and checks if the identities $\mathbf{C}$, $\mathbf{S}_L$ and SID'_c are decrypted correctly. If the identities are not correct, $\mathbf{S}_L$ rejects $\mathbf{C}$. Otherwise, $\mathbf{S}_L$ uses password $\mathbf{pw}_L$ to check if $v'_L = \mathbf{H}_1(\mathbf{pw}_L || \tau'_C || \bar{\tau}'_L || \text{SID}'_c)$ is correct. If the check is incorrect, $\mathbf{S}_L$ rejects $\mathbf{C}$. Otherwise, $\mathbf{S}_L$ uses new subgroup key K'_L computed in Phase 1 Round 3 to compute:

$$\begin{aligned} y'_L &= g^{\mathbf{H}_2(K'_L)} \bmod p, \quad (y'_L)^* = y'_L \cdot h^{\mathbf{H}_1(\text{SID}'_c || \mathbf{S}_L || \mathbf{C} || \mathbf{pw}_L || \tau'_C || \bar{\tau}'_L)} \\ \mu'_L &= \text{MAC}_{ak_L}(\tau'_C || \bar{\tau}'_L || (y'_L)^* || \text{SID}'_c || \mathbf{S}_L || \mathbf{C} || 00) \\ \bar{\mu}'_L &= \text{MAC}_{ak_L}(\tau'_C || \bar{\tau}'_L || y'_L || \text{SID}'_c || \mathbf{S}_L || \mathbf{C} || 01) \end{aligned}$$

Finally, $\mathbf{S}_L$ sends $\{\text{SID}'_c || \mathbf{S}_L || (y'_L)^* || \mu'_L || \bar{\mu}'_L\}$ to $\mathbf{C}$.

(ii) Subgroup controller $\mathbf{S}_i$ for $1 \leq i \neq L \leq n_c$ first decrypts c'_i and checks if the identities $\mathbf{C}$, $\mathbf{S}_i$ and SID'_c are decrypted correctly. If the identities are not correct, $\mathbf{S}_i$ rejects $\mathbf{C}$. Otherwise, $\mathbf{S}_i$ uses password $\mathbf{pw}_i$ to check if $v'_i = \mathbf{H}_1(\mathbf{pw}_i || \tau'_C || \bar{\tau}'_i || \text{SID}'_c)$ is correct. If the check is incorrect, $\mathbf{S}_i$ rejects $\mathbf{C}$. Otherwise, $\mathbf{S}_i$ stores τ'_C , $\bar{\tau}'_i$ and SID'_c for later computation.

Round 3: Main controller $\mathbf{C}$ first checks if $\mu'_L = \text{MAC}_{ak_L}(\tau'_C || \bar{\tau}'_L || (y'_L)^* || \text{SID}'_c || \mathbf{S}_L || \mathbf{C} || 00)$. If the

check is not correct, $\mathbf{C}$ blocks malicious subgroup controller $\mathbf{S}_L$. Otherwise, $\mathbf{C}$ obtains $y'_L = (y_L^*)' \cdot h^{-H_1(\text{SID}'_c || \mathbf{S}_L || \mathbf{C} || \text{pw}_L || \tau'_c || \bar{\tau}'_L)}$ and checks $\bar{\mu}'_L = \text{MAC}_{ak_L}(\tau'_c || \bar{\tau}'_L || y'_L || \text{SID}'_c || \mathbf{S}_L || \mathbf{C} || 01)$. If the check is incorrect, then $\mathbf{C}$ blocks malicious subgroup controller $\mathbf{S}_L$. Otherwise, $\mathbf{C}$ selects new random $r'_c \in Z_q^*$ and computes $y'_c = g^{r'_c} \bmod p$. Next, $\mathbf{C}$ computes the new DH key t'_L and t'_i that it shares with $\mathbf{S}_L$ and $\mathbf{S}_i$ for $1 \leq i \neq L \leq n_c$ respectively, the new final group key K' , θ'_c , $\hat{\mu}'_L$ and $\hat{\mu}'_i$ for $1 \leq i \neq L \leq n_c$:

$$\begin{aligned} t'_L &= (y'_L)^{r'_c} \bmod p, \quad t'_i = y_i^{r'_c} \bmod p \\ K' &= \left(y'_c \cdot y'_L \cdot \prod_{i=1, i \neq L}^{n_c} y_i \right)^{r'_c} \bmod p \\ \theta'_c &= H_1(\text{SID}'_c || \text{UID}_c || K' || \tau'_c || y'_c) \\ \hat{\mu}'_L &= \text{MAC}_{ak_L}(\tau'_c || \bar{\tau}'_L || y'_c || \theta'_c || \text{SID}'_c || \mathbf{C} || \mathbf{S}_L || 11) \\ \hat{\mu}'_i &= \text{MAC}_{ak_i}(\tau'_c || \bar{\tau}'_i || y'_c || \theta'_c || \text{SID}'_c || \mathbf{C} || \mathbf{S}_i || 11) \end{aligned}$$

where $\text{UID}_c = \mathbf{C} || \mathbf{S}_1 || \mathbf{S}_2 || \dots || \mathbf{S}_{n_c}$. $\mathbf{C}$ encrypts $K' || \mathbf{C} || y'_L || \text{SID}'_c$ using t'_L and $K' || \mathbf{C} || y_i || \text{SID}'_c$ using t'_i for $1 \leq i \neq L \leq n_c$ and obtains $\hat{c}'_L = \mathcal{E}_{t'_L}(K' || \mathbf{C} || y'_L || \text{SID}'_c)$ and $\hat{c}'_i = \mathcal{E}_{t'_i}(K' || \mathbf{C} || y_i || \text{SID}'_c)$ for $1 \leq i \neq L \leq n_c$ respectively. $\mathbf{C}$ then broadcast $\{\text{SID}'_c || \text{UID}_c || \mathbf{C} || y'_c || \hat{\mu}'_L || \hat{\mu}'_i || \hat{c}'_L || \hat{c}'_i || \theta'_c\}$ for $1 \leq i \neq L \leq n_c$.

Phase 3: Key Distribution

Round 1:

(i) $\mathbf{S}_L$ first checks if $\hat{\mu}'_L = \text{MAC}_{ak_L}(\tau'_c || \bar{\tau}'_L || y'_c || \theta'_c || \text{SID}'_c || \mathbf{C} || \mathbf{S}_L || 11)$ is correct. If the equation is incorrect, $\mathbf{S}_L$ rejects $\mathbf{C}$. Otherwise, $\mathbf{S}_L$ computes $t'_L = (y'_c)^{H_2(K'_L)} \bmod p$ and uses t'_L to decrypt $\hat{c}'_L$ and obtains the new final group key K' , identity $\mathbf{C}$, y'_L and SID'_c . If the identity $\mathbf{C}$, y'_L and SID'_c are not decrypted correctly, $\mathbf{S}_L$ rejects $\hat{c}'_L$. Otherwise, $\mathbf{S}_L$ confirms and accepts the new final group key K' if the key confirmation code $\theta'_c = H_1(\text{SID}'_c || \text{UID}_c || K' || \tau'_c || y'_c)$ holds. Next, $\mathbf{S}_L$ selects new random $\tau'_L \in Z_q^*$, $\tau'_{L,j} \in Z_q^*$ for $1 \leq j \leq n_L - \ell$ and session identity SID'_L , encrypts $\tau'_L || \tau'_{L,j} || \mathbf{S}_L || \mathbf{V}_{L,j} || \text{SID}'_L$ using $ek_{L,j}$ and obtains ciphertext $c'_{L,j} = \mathcal{E}_{ek_{L,j}}(\tau'_L || \tau'_{L,j} || \mathbf{S}_L || \mathbf{V}_{L,j} || \text{SID}'_L)$. $\mathbf{S}_L$ then computes $v'_{L,j} = H_1(\text{pw}_{L,j} || \tau'_L || \tau'_{L,j} || \text{SID}'_L)$. Next, $\mathbf{S}_L$ encrypts $K'_L || K' || \mathbf{S}_L || z_{L,j} || \text{SID}'_L$ using $t'_{L,j}$ of remaining users $\mathbf{V}_{L,j}$ for $1 \leq j \leq n_L - \ell$ and obtains $\hat{c}'_{L,j} = \mathcal{E}_{t'_{L,j}}(K'_L || K' || \mathbf{S}_L || z_{L,j} || \text{SID}'_L)$ for

$1 \leq j \leq n_L - \ell$. S_L then computes the key confirmation code θ'_L and $\hat{\mu}'_{L,j}$ for $1 \leq j \leq n_L - \ell$:

$$\begin{aligned}\theta'_L &= H_1(\text{SID}'_L || \text{UID}'_L || K'_L || K' || \tau'_L || z'_L) \\ \hat{\mu}'_{L,j} &= \text{MAC}_{ak_{L,j}}(\tau'_L || \tau'_{L,j} || z'_L || \theta'_L || \text{SID}'_L || S_L || V_{L,j} || 11)\end{aligned}$$

where $\text{UID}'_L = S_L || V_{L,1} || V_{L,2} || \dots || V_{L,n_L-\ell}$. Finally, S_L broadcast $\{\text{SID}'_L || \text{UID}'_L || S_L || c'_{L,j} || v'_{L,j} || z'_L || \hat{\mu}'_{L,j} || \hat{c}'_{L,j} || \theta'_L\}$ for $1 \leq j \leq n_L - \ell$ to remaining users $V_{L,j}$ for $1 \leq j \leq n_L - \ell$.

(ii) Each S_i for $1 \leq i \neq L \leq n_c$ first checks if $\hat{\mu}'_i = \text{MAC}_{ak_i}(\tau'_c || \tau'_i || y'_c || \theta'_c || \text{SID}'_c || C || S_i || 11)$ is correct. If the equation is incorrect, S_i rejects C . Otherwise, S_i computes $t'_i = (y'_c)^{H_2(K_i)} \bmod p$ and uses t'_i to decrypt $\hat{c}'_i$ and obtains the new final group key K' , identity C , y_i and SID'_c . If the identity C , y_i and SID'_c are not decrypted correctly, S_i rejects $\hat{c}'_i$. Otherwise, S_i confirms and accepts the new final group key K' if the key confirmation code $\theta'_c = H_1(\text{SID}'_c || \text{UID}_c || K' || \tau'_c || y'_c)$ holds. Next, S_i selects $\tau'_i \in Z_q^*$ and new session identity SID'_i and encrypts $K' || S_i || \text{SID}'_i || \tau'_i$ using K_i and obtains $\hat{c}''_i = \mathcal{E}_{K_i}(K' || S_i || \text{SID}'_i || \tau'_i)$. S_i then computes the key confirmation code $\theta'_i = H_1(\text{SID}'_i || \text{UID}_i || K' || \tau'_i || z_i)$, where $\text{UID}_i = S_i || U_{i,1} || U_{i,2} || \dots || U_{i,n_i}$. Finally, S_i broadcast $\{\text{SID}'_i || \text{UID}_i || S_i || \hat{c}''_i || \theta'_i\}$ to subgroup $\mathcal{G}_i$.

Round 2:

(i) Each remaining user $V_{L,j}$ for $1 \leq j \leq n_L - \ell$ first decrypts $c'_{L,j}$ and checks if the identities S_L , $V_{L,j}$ and SID'_L are decrypted correctly. If the identities are not correct, $V_{L,j}$ rejects S_L . Otherwise, $V_{L,j}$ uses password $\text{pw}_{L,j}$ to check the validity of $v'_{L,j} = H_1(\text{pw}_{L,j} || \tau'_L || \tau'_{L,j} || \text{SID}'_L)$. If it is incorrect, $V_{L,j}$ rejects S_L . Otherwise, $V_{L,j}$ checks if $\hat{\mu}'_{L,j} = \text{MAC}_{ak_{L,j}}(\tau'_L || \tau'_{L,j} || z'_L || \theta'_L || \text{SID}'_L || S_L || V_{L,j} || 11)$ is valid. If the equation is invalid, $V_{L,j}$ rejects the malicious subgroup controller S_L . Otherwise, $V_{L,j}$ computes $t'_{L,j} = (z'_L)^{r_{L,j}} \bmod p$ and uses $t'_{L,j}$ to decrypt $\hat{c}'_{L,j}$ and obtains new subgroup key K'_L , new final group key K' , identity S_L , $z_{L,j}$ and SID'_L . If the identity S_L , $z_{L,j}$ and SID'_L are not decrypted correctly, then $V_{L,j}$ rejects $\hat{c}'_{L,j}$. Otherwise, $V_{L,j}$ confirms and accepts both subgroup key K'_L and final group key K' if the key confirmation code $\theta'_L = H_1(\text{SID}'_L || \text{UID}'_L || K'_L || K' || \tau'_L)$ holds.

(ii) Each $U_{i,j}$ for $1 \leq i \neq L \leq n_c$ and $1 \leq j \leq n_i$ uses K_i to decrypt $\hat{c}''_i$ and obtains new final group key K' , identity S_i , SID'_i and τ'_i . If the identity S_i and SID'_i are not decrypted correctly, then $U_{i,j}$ rejects $\hat{c}''_i$. Otherwise, $U_{i,j}$ confirms and accepts new final group key K' if the key confirmation code $\theta'_i = H_1(\text{SID}'_i || \text{UID}_i || K' || \tau'_i || z_i)$ holds.

10.3.1 Security Analysis

In this section, we provide the security analysis of the three proposed password-based GKA protocols, namely Initial Group Key Agreement (IKA), Join and Leave Protocol. We show that the proposed protocols are secure as follows:

Security Analysis of IKA Protocol

As the proposed Initial GKA (IKA) protocol (Section 10.3) is built from a STAR structure (where there is a main controller that communicates with all the user nodes) which is similar to VSTAR protocol (Section 9.4.1) with slight differences, we call it a ESTAR protocol. Before we describe the ESTAR protocol, we note that $H(x, m)$ is a cryptographic hash function where x is a secure element (which consists of password and element being encrypted) and m is a public element (which is different for each protocol run). We also assume that x is known to user U and controller C . The ESTAR protocol is described as follows:

Round 1: Each user U_i ($1 \leq i \leq n$) chooses a random $r_i \in Z_q^*$ and m_i is a publicly known information, computes $z_i^* = g^{r_i} h^{H(x_i, m_i)}$ and sends z_i^* and m_i .

Round 2: Upon receipt of all z_i^* for $1 \leq i \leq n$, controller C chooses a random $r_c \in Z_q^*$ and computes $z_c = g^{r_c}$, $T_i = (z_i^* h^{-H(x_i, m_i)})^{r_c}$ for $1 \leq i \leq n$ (It is noted that x_i is known to user U_i and controller C) and the group key $K = z_c^{r_c} \cdot \prod_{i=1}^n T_i = g^{r_c r_c + \sum_{i=1}^n r_c r_i}$. Next, C encrypts K as $c_i = \mathcal{E}_{T_i}(K)$ for $1 \leq i \leq n$ and sends out z_c and c_i for $1 \leq i \leq n$.

Computation of Key: Upon receipt of z_c and c_i , each user U_i ($1 \leq i \leq n$) computes $T_i = z_c^{r_i}$ and obtains the group key K by decrypting c_i .

We state the security of the ESTAR protocol as follows:

Theorem 41 *If the VSTAR protocol is secure, then the ESTAR protocol is secure.*

Proof: : Since $g^{r_i} h^{H(x_i, m_i)}$ for $1 \leq i \leq n$ and g^{r_c} are sent in round 1 and round 2 respectively, therefore, g^{r_i} can be viewed as “encrypted” by $h^{H(x_i, m_i)}$ and g^{r_c} is a public information. If an adversary knows x_i for all $1 \leq i \leq n$, then the adversary obtains all g^{r_i} and this is in fact same as VSTAR protocol. By the assumption that the VSTAR protocol is secure, therefore, the ESTAR

protocol is secure in this case. If the adversary does not know all x_i , then $g^{r_i} h^{H(x_i, m_i)}$ and g^{r_i} will look random to each other in the adversary's view as $h^{H(x_i, m_i)}$ is random. So, there is no information given to the adversary about g^{r_i} . Therefore, the group key K and the public information $g^{r_i} h^{H(x_i, m_i)}$ and g^{r_c} are appeared to be random in the adversary's view. Hence, we conclude that the ESTAR protocol is secure. ■

Next, we prove that the proposed password-based IKA protocol is secure against off-line and on-line password guessing.

Theorem 42 *The proposed password-based IKA protocol is secure against off-line and on-line password guessing attack if the symmetric-key encryption scheme SKE and the message authentication code scheme MACS are secure.*

Proof: Since a password is used in Round 1 and 2 of Phase 1 and in Round 1 and 2 of Phase 2 respectively, and furthermore, the functionality of Round 1 and Round 2 of Phase 1 are the same as that of Phase 2 respectively. The only difference between the two phases are the communication between the subgroup controller S_i and users $U_{i,j}$ in Phase 1 and the communication between the main controller C and subgroup controllers S_i in Phase 2. Hence, we only consider the off-line and on-line password guessing attack on Round 1 and Round 2 in Phase 1. First, we consider an adversary $\mathcal{A}$ launching an off-line password guessing attack on the user $U_{i,j}$. Since τ_i and $\tau_{i,j}$ is encrypted using encryption key $ek_{i,j}$ which is secret and the symmetric encryption scheme SKE is secure, hence, τ_i and $\tau_{i,j}$ is unknown to an adversary. Therefore, it is not possible for the adversary $\mathcal{A}$ to verify the correct guess of password from $v_{i,j}$. Furthermore, MACS is secure and τ_i and $\tau_{i,j}$ is unknown to the adversary, therefore, it is not possible for the adversary $\mathcal{A}$ to verify the correct guess of $z_{i,j}$ from $\bar{\mu}_{i,j}$ through the password guessing in $z_{i,j}^*$. Hence, the proposed password-based IKA protocol is secure against off-line password guessing.

Next, we consider an adversary $\mathcal{A}$ launching an on-line password guessing attack on the user $U_{i,j}$. First, consider an adversary $\mathcal{A}$ randomly generating $\hat{\tau}_i$ and $\hat{\tau}_{i,j}$ and computing $\hat{v}_{i,j}$ by each guess of password. As the adversary $\mathcal{A}$ does not know the encryption key $ek_{i,j}$, therefore, $\mathcal{A}$ can only randomly generate $\hat{c}_{i,j}$ and send it to user $U_{i,j}$. Then, $U_{i,j}$ decrypts $\hat{c}_{i,j}$ to obtain τ'_i and $\tau'_{i,j}$ which is not equal to $\hat{\tau}_i$ and $\hat{\tau}_{i,j}$ respectively, and compute $v'_{i,j}$ which is not equal to $\hat{v}_{i,j}$. Therefore,

$U_{i,j}$ always rejects the request. Secondly, consider the adversary $\mathcal{A}$ has obtained a valid $v_{i,j}$ and $c_{i,j}$ from the subgroup controller S_i in Round 1 of Phase 1. The adversary $\mathcal{A}$ randomly generate $z_{i,j}$, $z_{i,j}^*$ and $\mu_{i,j}$ as the adversary $\mathcal{A}$ does not know the authentication key $ak_{i,j}$ and encryption key $ek_{i,j}$. Upon receipt of $\mu_{i,j}$, the subgroup controller S_i will reject the request. In this case, it is not possible for $\mathcal{A}$ to obtain the correct guess of the password. Hence, the proposed password-based IKA protocol is secure against on-line password guessing attack on a user $U_{i,j}$.

The case of launching an off-line and on-line password guessing attack on the subgroup controller's (S_i) password pw_i is same as above. Therefore, we omit the proof here. ■

Next, We show that an adversary is not able to obtain subgroup keys K_i for $1 \leq i \leq n_c$ and final group key K from the proposed password-based IKA protocol.

Theorem 43 *An adversary $\mathcal{A}$ is not able to obtain the subgroup keys K_i and final group key K in the proposed IKA protocol if the ESTAR protocol, the symmetric key encryption scheme SKE and the message authentication code scheme MACS are secure.*

Proof: In this proof, we consider two cases, that is (a) the adversary $\mathcal{A}$ tries to impersonate user $U_{i,j}$, subgroup controller S_i or main controller C to obtain subgroup keys K_i for $1 \leq i \leq n_c$ and final group key K and (b) the adversary $\mathcal{A}$ tries to obtain subgroup keys K_i and final group key K from the public information sent in the IKA protocol. For case (a), we first consider $\mathcal{A}$ impersonating a user $U_{i,j}$. Then in Phase 1 Round 2, $\mathcal{A}$ computes $z_{i,j} = g^{r_{i,j}}$. Next, $\mathcal{A}$ has to compute valid $z_{i,j}^* = z_{i,j} \cdot h^{\text{H}_1(\text{SID}_i || U_{i,j} || S_i || pw_{i,j} || \tau_i || \tau_{i,j})}$, $\mu_{i,j} = \text{MAC}_{ak_{i,j}}(\tau_i || \tau_{i,j} || z_{i,j}^* || \text{SID}_i || U_{i,j} || S_i || 00)$ and $\bar{\mu}_{i,j} = \text{MAC}_{ak_{i,j}}(\tau_i || \tau_{i,j} || z_{i,j} || \text{SID}_i || U_{i,j} || S_i || 01)$. The validity of $z_{i,j}^*$, $\mu_{i,j}$ and $\bar{\mu}_{i,j}$ are checked by S_i in Phase 1 Round 3. If either one of $z_{i,j}^*$, $\mu_{i,j}$ or $\bar{\mu}_{i,j}$ is not valid, then S_i rejects invalid user $U_{i,j}$. As $\mathcal{A}$ does not know the encryption key $ek_{i,j}$ of $U_{i,j}$ and the symmetric-key encryption scheme SKE is assumed to be secure, therefore $\mathcal{A}$ is not able to obtain τ_i and $\tau_{i,j}$ from $c_{i,j} = \mathcal{E}_{ek_{i,j}}(\tau_i || \tau_{i,j} || S_i || U_{i,j} || \text{SID}_i)$. Furthermore, $\mathcal{A}$ does not know the password $pw_{i,j}$ of $U_{i,j}$ and from Theorem 42, it is shown that $\mathcal{A}$ is not able to obtain password $pw_{i,j}$ through off-line and on-line password guessing attacks, hence $\mathcal{A}$ is not able to compute the valid $z_{i,j}^*$. As $\mathcal{A}$ does not know the authentication key $ak_{i,j}$ of $U_{i,j}$ and the message authentication code scheme MACS is assumed to be secure, $\mathcal{A}$ is also not able to produce a valid $\mu_{i,j}$ and $\bar{\mu}_{i,j}$. Hence, it is not possible for the adversary $\mathcal{A}$ to impersonate user $U_{i,j}$.

Next consider $\mathcal{A}$ impersonating a subgroup controller, say S_i , in Phase 1. Then in Phase 1 Round 1, $\mathcal{A}$ selects random τ_i and $\tau_{i,j}$ and session identity SID_i . Next, $\mathcal{A}$ has to compute the valid $c_{i,j} = \mathcal{E}_{ek_{i,j}}(\tau_i || \tau_{i,j} || S_i || U_{i,j} || SID_i)$ and $v_{i,j} = H_1(\text{pw}_{i,j} || \tau_i || \tau_{i,j} || SID_i)$ for $1 \leq j \leq n_i$. The validity of $c_{i,j}$ and $v_{i,j}$ are checked by $U_{i,j}$ in Phase 1 Round 2. If either $c_{i,j}$ or $v_{i,j}$ is not valid, then $U_{i,j}$ rejects the invalid S_i . As $\mathcal{A}$ does not know the encryption key $ek_{i,j}$ of $U_{i,j}$ and the symmetric-key encryption scheme SKE is assumed to be secure, therefore $\mathcal{A}$ is not able to produce a valid $c_{i,j}$. $\mathcal{A}$ also does not know the password $\text{pw}_{i,j}$ of $U_{i,j}$, therefore $\mathcal{A}$ is not able to produce a valid $v_{i,j}$. Hence $\mathcal{A}$ is not able to impersonate S_i in Phase 1.

Next, consider $\mathcal{A}$ impersonating S_i in Phase 2. As the functionality of Round 1 and Round 2 of Phase 1 are same as that of Round 1 and Round 2 of Phase 2 respectively and the difference between the two phases are the communications between S_i and $U_{i,j}$ in Phase 1 and the communication between C and S_i in Phase 2, hence the proof of $\mathcal{A}$ impersonating S_i in Phase 2 is similar to the impersonation of $U_{i,j}$ in Phase 1. Hence, we omit the proof here.

Next consider $\mathcal{A}$ impersonating S_i in Phase 3 Round 1, then $\mathcal{A}$ needs to compute valid $\hat{c}_{i,j} = \mathcal{E}_{t_{i,j}}(K_i || K || S_i || z_{i,j} || SID_i)$, $\hat{\mu}_{i,j} = \text{MAC}_{ak_{i,j}}(\tau_i || \tau_{i,j} || z_i || \theta_i || SID_i || S_i || U_{i,j} || 11)$ and $\theta_i = H_1(SID_i || UID_i || K_i || K || \tau_i || z_i)$. The validity of $\hat{c}_{i,j}$ and $\hat{\mu}_{i,j}$ are check by $U_{i,j}$ in Phase 3 Round 2. If either one of $\hat{c}_{i,j}$ or $\hat{\mu}_{i,j}$ is not correct, then $U_{i,j}$ rejects invalid S_i . As $\mathcal{A}$ does not know the authentication key $ak_{i,j}$ and the message authentication code scheme MACS is assumed to be secure, therefore $\mathcal{A}$ is not able to compute a valid $\hat{\mu}_{i,j}$. Furthermore, $\mathcal{A}$ does not know password $\text{pw}_{i,j}$ of $U_{i,j}$, hence $\mathcal{A}$ is not able to obtain $z_{i,j}$ from $z_{i,j}^*$ to compute valid $t_{i,j} = g^{r_i r_{i,j}}$. Therefore, $\mathcal{A}$ is not able to compute the ciphertext $\hat{c}_{i,j}$. Hence $\mathcal{A}$ is not able to impersonate S_i in Phase 3.

Finally, consider $\mathcal{A}$ impersonating main controller C in Phase 2 Round 1. The proof is similar to the impersonation of a subgroup controller S_i in Phase 1 Round 1. Hence we omit the proof here. Next consider $\mathcal{A}$ impersonating main controller C in Phase 2 Round 3. Then $\mathcal{A}$ needs to compute valid $\hat{c}_i = \mathcal{E}_{t_i}(K || C || y_i || SID_c)$, $\hat{\mu}_i = \text{MAC}_{ak_i}(\tau_c || \bar{\tau}_i || y_c || \theta_c || SID_c || C || S_i || 11)$ and $\theta_c = H_1(SID_c || UID_c || K || \tau_c || y_c)$. The validity of $\hat{c}_i$ and $\hat{\mu}_i$ are checked by S_i in Phase 3 Round 1. If either one of $\hat{c}_i$ or $\hat{\mu}_i$ is not correct, then S_i rejects invalid C . As $\mathcal{A}$ does not know password pw_i of S_i , hence $\mathcal{A}$ is not able to obtain y_i from y_i^* to compute valid $t_i = g^{r_c H_2(K_i)}$. Therefore, $\mathcal{A}$ is not able to compute the ciphertext $\hat{c}_i$. Furthermore, $\mathcal{A}$ does not know the authentication key ak_i and

the message authentication code scheme MACS is assumed to be secure, therefore $\mathcal{A}$ is not able to compute a valid $\hat{\mu}_i$. Hence, $\mathcal{A}$ is not able to impersonate $\mathcal{C}$ in Phase 2 Round 3. Therefore, it is not possible for $\mathcal{A}$ to obtain subgroup key K_i and final group key K in case (a).

Now we consider case (b) as follows: First, we consider $\mathcal{A}$ tries to obtain subgroup key K_i from the public information sent in the IKA protocol. But by ignoring the authentication information $(\mu_{i,j}, \bar{\mu}_{i,j}, \hat{\mu}_{i,j} \text{ and } \theta_i)$ sent, the proposed IKA protocol to form the subgroup key K_i is just the ESTAR protocol, where Phase 1 Round 2, Phase 1 Round 3 together with Phase 3 Round 1, and Phase 3 Round 2 of the IKA protocol is just Round 1, Round 2 and Computation of Key of the ESTAR protocol respectively. By Theorem 41, it is shown that the ESTAR protocol is secure. Hence, $\mathcal{A}$ is not able to obtain the subgroup key K_i from the proposed IKA protocol.

Next, we consider $\mathcal{A}$ tries to obtain final group key K from the public information sent in IKA protocol. But by ignoring the authentication information $(\mu_i, \bar{\mu}_i, \hat{\mu}_i \text{ and } \theta_c)$ sent, the proposed IKA protocol to form the final group key K is just the ESTAR protocol, where Phase 2 Round 2, Phase 2 Round 3 and Phase 3 Round 1 of the IKA protocol is just Round 1, Round 2 and Computation of Key of the ESTAR protocol respectively. By Theorem 41, it is shown that the ESTAR protocol is secure. Hence, $\mathcal{A}$ is not able to obtain the final group key K . The final group key K is also the plaintext of $\hat{c}_{i,j} = \mathcal{E}_{t_{i,j}}(K_i || K || S_i || z_{i,j} || SID_i)$. As $\hat{c}_{i,j}$ is part of the information sent in the ESTAR protocol to establish the subgroup key K_i and both K_i and K are the plaintext of $\hat{c}_{i,j}$, hence $\mathcal{A}$ is also not able to obtain final group key K from $\hat{c}_{i,j}$ as the ESTAR protocol is secure. Therefore, $\mathcal{A}$ is not able to obtain the final group key K from the proposed password-based IKA protocol. ■

Next, we prove that an adversary is not able to launch a DoS attack against the proposed password-based IKA protocol. The theorem is stated as follows:

Theorem 44 *Assume that the computation of the hash functions H_1 and H_2 , the message authentication code scheme MACS and the symmetric-key encryption scheme SKE is efficient, then the proposed password-based IKA protocol is resistant to DoS attacks.*

Proof: The purpose of launching a DoS attack on the password-based IKA protocol is to prevent all the legitimate users $U_{i,j}$, subgroup controllers S_i and main controller $\mathcal{C}$ to establish a final group key. In order to do this, an adversary can launch the attack on one of the subgroup controller or main controller. Now, consider an adversary that tries to attack a subgroup controller,

say S_i . First consider the adversary sending $\mu_{i,j}$ and $\bar{\mu}_{i,j}$ in Round 2 of Phase 1 to the subgroup controller S_i . As the adversary does not know the authentication key $ak_{i,j}$ and password $pw_{i,j}$, the adversary will not be able to compute a correct $\mu_{i,j}$ and $\bar{\mu}_{i,j}$. Therefore, the subgroup controller S_i can easily detect an invalid $\mu_{i,j}$ by the efficient computation of MACS¹. Hence, the subgroup controller S_i can immediately reject the invalid user $U_{i,j}$. The adversary is also not able to replay previously sent $\mu_{i,j}$ and $\bar{\mu}_{i,j}$ because the τ_i , $\tau_{i,j}$ and SID_i sent by S_i are fresh for every protocol run. The other way to launch a DoS attack on the subgroup controller S_i is by invalid main controller. Now, the adversary sent v_i and c_i to the subgroup controller S_i in Round 1 of Phase 2. As the adversary does not know the encryption key ek_i and the password pw_i , the adversary will not be able to compute the correct v_i and c_i . Hence, the subgroup controller S_i can easily detect the invalidity of v_i and c_i by the efficient computation of symmetric-key encryption scheme and hash function¹. Hence, the adversary is not able to launch DoS attacks on subgroup controller S_i .

Now, consider an adversary that tries to launch a DoS attack on the main controller C . First, the adversary tries to send μ_i to the main controller in Round 2 of Phase 2. As the adversary does not know the encryption key ek_i and authentication key ak_i , the adversary is not able to compute the correct μ_i . Hence, the main controller C is able to detect the invalidity of μ_i by the efficient computation of message authentication code and immediately reject the sender S_i . In this case, the adversary is not able to launch DoS attack on the main controller. The adversary is also not able to replay old messages in Round 1 and Round 2 of Phase 2 because the τ_c , $\bar{\tau}_i$ and SID_c sent by C in Round 1 of Phase 2 are fresh for every protocol run. ■

Security Analysis of Join Protocol

Theorem 45 *The proposed password-based Join protocol achieves backward secrecy provided the CDH problem is difficult.*

Proof: First, consider the new user $U_{J,k}$ tries to obtain the previous subgroup key $K_J = g^{r_J r_J + \sum_{j=1}^{n_J} r_J r_{J,j}}$ from the new subgroup key $K'_J = g^{r'_J r'_J + \sum_{j=1}^{n_J} r'_J r_{J,j} + \sum_{k=n_J+1}^{n_J+m} r'_J r'_{J,k}}$. As r'_J and r_J

¹From the Crypto++ 5.5 Library benchmarks [Cry] on the Pentium 4 2.13GHz microprocessor, it only takes 247.8 nanoseconds for a 160-bit HMAC(SHA-1) MAC computation, 250.6 nanoseconds for a 160-bit SHA-1 Hash computation and 12.63 nanoseconds per byte for the AES/CBC (128-bit key) encryption algorithm. In comparison, RSA 1024-bit signature generation takes 3.02 milliseconds per operation and RSA 1024-bit signature verification takes 0.13 milliseconds per operation using 17 as the public exponent on the same system. Hence the MAC algorithm, hash function and symmetric key encryption scheme are efficient.

are random and independent to each other, therefore, $g^{r'_J r'_J}$ and $g^{r_J r_J}$ are independent to each other and $g^{r'_J r_{J,j}}$ and $g^{r_J r_{J,j}}$ for $1 \leq j \leq n_J$ are independent to each other. Furthermore, the terms $g^{r'_J r'_{J,k}}$ for $n_J + 1 \leq k \leq n_J + m$ are not in the previous subgroup key K_J , therefore K'_J and K_J are independent to each other. If $U_{J,k}$ can obtain $g^{r_J r_J}$ from $z_J = g^{r_J}$ and $g^{r_J r_{J,j}}$ from $z_J = g^{r_J}$ and $z_{J,j} = g^{r_{J,j}}$ for $1 \leq j \leq n_J$ respectively, then $U_{J,k}$ can obtain the previous subgroup key $K_J = g^{r_J r_J + \sum_{j=1}^{n_J} r_J r_{J,j}}$. But this is not possible for $U_{J,k}$ to obtain $g^{r_J r_J}$ from z_J and $g^{r_J r_{J,j}}$ from z_J and $z_{J,j}$ respectively as this is the CDH problem and is assumed to be difficult. Furthermore, $U_{J,k}$ is not able to obtain $z_{J,j}$ from $z_{J,j}^* = z_{J,j} \cdot h^{\text{H}_1(\text{SID}_J || U_{J,j} || S_J || \text{pw}_{J,j} || \tau_J || \tau_{J,j})}$ as $U_{J,k}$ does not know the password $\text{pw}_{J,j}$ of user $U_{J,j}$ for $1 \leq j \leq n_J$. Hence, new user $U_{J,k}$ is not able to obtain previous subgroup key K_J .

Next, consider the new user $U_{J,k}$ tries to obtain previous final group key $K = g^{r_c r_c + \sum_{i=1}^{n_c} r_c \text{H}_2(K_i)}$ from the new group key $K' = g^{r'_c r'_c + r'_c \text{H}_2(K'_J) + \sum_{i=1, i \neq J}^{n_c} r'_c \text{H}_2(K_i)}$. As r'_c and r_c are random and independent to each other, therefore, $g^{r'_c r'_c}$ and $g^{r_c r_c}$ are independent to each other and $g^{r'_c \text{H}_2(K_i)}$ and $g^{r_c \text{H}_2(K_i)}$ for $1 \leq i \neq J \leq n_c$ are independent to each other. Furthermore, as K'_J and K_J are random and independent to each other, therefore $\text{H}_2(K'_J)$ and $\text{H}_2(K_J)$ are independent to each other as the output of the hash function H_2 is independent. Hence, $g^{r'_c \text{H}_2(K'_J)}$ and $g^{r_c \text{H}_2(K_J)}$ are also independent to each other. Hence, K' and K are independent to each other. If $U_{J,k}$ can obtain $g^{r_c r_c}$ from $y_c = g^{r_c}$ and $g^{r_c \text{H}_2(K_i)}$ from $y_c = g^{r_c}$ and $y_i = g^{\text{H}_2(K_i)}$ for $1 \leq i \leq n_c$ respectively, then $U_{J,k}$ can obtain the previous group key $K = g^{r_c r_c + \sum_{i=1}^{n_c} r_c \text{H}_2(K_i)}$. But this is not possible for $U_{J,k}$ to obtain $g^{r_c r_c}$ from y_c and $g^{r_c \text{H}_2(K_i)}$ from y_c and y_i respectively as this is the CDH problem and is assumed to be difficult. Furthermore, $U_{J,k}$ is not able to obtain y_i from $y_i^* = y_i \cdot h^{\text{H}_1(\text{SID}_c || S_i || C || \text{pw}_i || \tau_c || \tau_i)}$ as $U_{J,k}$ does not know the password pw_i of S_i for $1 \leq i \leq n_c$. Therefore, $U_{J,k}$ is not able to obtain previous group key K . Hence, the proposed Join Protocol achieves backward secrecy. ■

The Join protocol also has to ensure that the adversary who eavesdrops on the communication cannot obtain new subgroup key K'_J and new final group key K' from the Join Protocol.

Theorem 46 *An adversary $\mathcal{A}$ is not able to obtain the new subgroup key K'_J and new final group key K' from the proposed password-based Join Protocol provided the ESTAR protocol, symmetric key encryption scheme SKE and the message authentication code scheme MACS are secure.*

Proof: In this proof, we consider 2 cases, that is, (a) the adversary $\mathcal{A}$ tries to impersonate the

subgroup controller S_J , subgroup controller S_i for $1 \leq i \neq J \leq n_c$, new users $U_{J,k}$ for $n_J + 1 \leq k \leq n_J + m$ or main controller C to establish the new subgroup key K'_J and new final group key K' (we do not consider $\mathcal{A}$ tries to impersonate users $U_{i,j}$ for $1 \leq i \neq J \leq n_c$ and $1 \leq j \leq n_i$ and $U_{J,j}$ for $1 \leq j \leq n_J$ as $U_{i,j}$ and $U_{J,j}$ does not send any information in the Join Protocol) and (b) the adversary $\mathcal{A}$ tries to obtain the new subgroup key K'_J and final group key K' from the public information sent in the Join Protocol. For case (a), we first consider $\mathcal{A}$ impersonating S_J in the Join Protocol. The proof that $\mathcal{A}$ is not able to impersonate S_J in Phase 1 and Phase 2 of the Join Protocol is the same as the proof that $\mathcal{A}$ is not able to impersonate S_i in Phase 1 and Phase 2 of the IKA protocol as shown in Theorem 43. Hence we omit the proof here. Next, consider $\mathcal{A}$ impersonating S_J in Phase 3 Round 1, then $\mathcal{A}$ needs to compute valid $\hat{c}'_{J,k} = \mathcal{E}_{t'_{J,k}}(K'_J || K' || S_J || z'_{J,k} || \text{SID}'_J)$, $\hat{c}''_J = \mathcal{E}_{K_J}(K'_J || K' || S_J || \text{SID}'_J || \tau'_J)$, $\hat{\mu}'_{J,k} = \text{MAC}_{ak_{J,k}}(\tau'_J || \tau'_{J,k} || z'_J || \theta'_J || \text{SID}'_J || S_J || U_{J,k} || 11)$ and $\theta'_J = H_1(\text{SID}'_J || \text{UID}'_J || K'_J || K' || \tau'_J || z'_J)$. The validity of $\hat{c}'_{J,k}$ and $\hat{\mu}'_{J,k}$ are checked by $U_{J,k}$ in Phase 3 Round 2 and the validity of $\hat{c}''_J$ is checked by $U_{J,j}$ in Phase 3 Round 2. If either one of $\hat{c}'_{J,k}$ or $\hat{\mu}'_{J,k}$ is not correct, then $U_{J,k}$ rejects invalid S_J . If $\hat{c}''_J$ is not correct, then $U_{J,j}$ rejects invalid S_J . As $\mathcal{A}$ does not know the authentication key $ak_{J,k}$ and the message authentication code scheme MACS is assumed to be secure, therefore $\mathcal{A}$ is not able to compute a valid $\hat{\mu}'_{J,k}$. Furthermore, $\mathcal{A}$ does not know password $\text{pw}_{J,k}$ of $U_{J,k}$, hence $\mathcal{A}$ is not able to obtain $z'_{J,k}$ from $(z^*_{J,k})'$ to compute valid $t'_{J,k} = g^{r'_J r'_{J,k}}$. Therefore, $\mathcal{A}$ is not able to compute the valid ciphertext $\hat{c}'_{J,k}$ as the symmetric key encryption scheme SKE is secure. As $\mathcal{A}$ does not know the existing subgroup key K_J and the symmetric key encryption scheme SKE is secure, therefore $\mathcal{A}$ is not able to compute the valid ciphertext $\hat{c}''_J$. Hence $\mathcal{A}$ is not able to impersonate S_J in Phase 3. Therefore, $\mathcal{A}$ is not able to impersonate S_J in the Join Protocol.

Next, we consider $\mathcal{A}$ wishes to impersonate subgroup controller S_i for $1 \leq i \neq J \leq n_c$ in the Join Protocol. As S_i for $1 \leq i \neq J \leq n_c$ does not send any information in Phase 1 and 2, hence we only consider $\mathcal{A}$ tries to impersonate S_i in Phase 3. Then in Phase 3 Round 1, $\mathcal{A}$ has to generate valid $\hat{c}''_i = \mathcal{E}_{K_i}(K' || S_i || \text{SID}'_i || \tau'_i)$ and $\theta'_i = H_1(\text{SID}'_i || \text{UID}'_i || K' || \tau'_i || z_i)$. The validity of $\hat{c}''_i$ is checked by $U_{i,j}$ for $1 \leq i \neq J \leq n_c$ and $1 \leq j \leq n_i$ in Phase 3 Round 2. If $\hat{c}''_i$ is not correct, then $U_{i,j}$ rejects invalid S_i . As $\mathcal{A}$ does not know the subgroup key K_i and the symmetric key encryption scheme SKE is secure, therefore $\mathcal{A}$ is not able to compute the valid ciphertext $\hat{c}''_i$. Hence, $\mathcal{A}$ is not able to impersonate S_i in Phase 3 of the Join Protocol.

Next, consider if $\mathcal{A}$ tries to impersonate new user $U_{J,k}$ for $n_J + 1 \leq k \leq n_J + m$. The proof of this is similar to the impersonation of users $U_{i,j}$ in Theorem 43. Hence we omit the proof here.

Next, we consider $\mathcal{A}$ tries to impersonate $\mathbf{C}$. Then in Phase 2 Round 1, $\mathcal{A}$ selects random $\tau'_C \in Z_q^*$, $\bar{\tau}'_J \in Z_q^*$, $\bar{\tau}'_i \in Z_q^*$ for $1 \leq i \neq J \leq n_c$ and session identity SID'_C . Next, $\mathcal{A}$ has to compute the valid $c'_J = \mathcal{E}_{ek_J}(\tau'_C || \bar{\tau}'_J || \mathbf{C} || \mathbf{S}_J || SID'_C)$, $v'_J = H_1(\text{pw}_J || \tau'_C || \bar{\tau}'_J || SID'_C)$, $c'_i = \mathcal{E}_{ek_i}(\tau'_C || \bar{\tau}'_i || \mathbf{C} || \mathbf{S}_i || SID'_C)$ and $v'_i = H_1(\text{pw}_i || \tau'_C || \bar{\tau}'_i || SID'_C)$ for $1 \leq i \neq J \leq n_c$. The validity of c'_J and v'_J (c'_i and v'_i) are checked by $\mathbf{S}_J$ ($\mathbf{S}_i$) in Phase 2 Round 2. If either c'_J or v'_J (c'_i or v'_i) is not valid, then $\mathbf{S}_J$ ($\mathbf{S}_i$) rejects the invalid $\mathbf{C}$. As $\mathcal{A}$ does not know the encryption key ek_J and ek_i of $\mathbf{S}_J$ and $\mathbf{S}_i$ respectively and the symmetric-key encryption scheme **SKE** is assumed to be secure, therefore $\mathcal{A}$ is not able to produce valid c'_J and c'_i . $\mathcal{A}$ also does not know password pw_J and pw_i of $\mathbf{S}_J$ and $\mathbf{S}_i$ respectively, therefore $\mathcal{A}$ is not able to produce valid v'_J and v'_i . Hence $\mathcal{A}$ is not able to impersonate $\mathbf{C}$ in Phase 2 Round 1. Next consider $\mathcal{A}$ impersonating main controller $\mathbf{C}$ in Phase 2 Round 3. Then $\mathcal{A}$ needs to compute valid $\hat{c}'_J = \mathcal{E}_{t'_J}(K' || \mathbf{C} || y'_J || SID'_C)$, $\hat{\mu}'_J = \text{MAC}_{ak_J}(\tau'_C || \bar{\tau}'_J || y'_C || \theta'_C || SID'_C || \mathbf{C} || \mathbf{S}_J || 11)$, $\hat{c}'_i = \mathcal{E}_{t'_i}(K' || \mathbf{C} || y_i || SID'_C)$, $\hat{\mu}'_i = \text{MAC}_{ak_i}(\tau'_C || \bar{\tau}'_i || y'_C || \theta'_C || SID'_C || \mathbf{C} || \mathbf{S}_i || 11)$ for $1 \leq i \neq J \leq n_c$ and $\theta'_C = H_1(SID'_C || \text{UID}_C || K' || \tau'_C || y'_C)$. The validity of $\hat{c}'_J$ and $\hat{\mu}'_J$ ($\hat{c}'_i$ and $\hat{\mu}'_i$) are checked by $\mathbf{S}_J$ ($\mathbf{S}_i$) in Phase 3 Round 1. If either one of $\hat{c}'_J$ or $\hat{\mu}'_J$ ($\hat{c}'_i$ or $\hat{\mu}'_i$) is not correct, then $\mathbf{S}_J$ ($\mathbf{S}_i$) rejects invalid $\mathbf{C}$. As $\mathcal{A}$ does not know password pw_J and pw_i of $\mathbf{S}_J$ and $\mathbf{S}_i$ for $1 \leq i \neq J \leq n_c$ respectively, hence $\mathcal{A}$ is not able to obtain y'_J from $(y_J^*)'$ and y_i from y_i^* to compute valid $t'_J = g^{r'_C H_2(K'_J)}$ and $t'_i = g^{r'_C H_2(K_i)}$ respectively. Therefore, $\mathcal{A}$ is not able to compute the valid ciphertext $\hat{c}'_J$ and $\hat{c}'_i$ for $1 \leq i \neq J \leq n_c$ as the symmetric-key encryption scheme **SKE** is assumed to be secure. Furthermore, $\mathcal{A}$ does not know the authentication key ak_J and ak_i for $1 \leq i \neq J \leq n_c$ and the message authentication code scheme **MACS** is assumed to be secure, therefore $\mathcal{A}$ is not able to compute a valid $\hat{\mu}'_J$ and $\hat{\mu}'_i$ respectively. Hence, $\mathcal{A}$ is not able to impersonate $\mathbf{C}$ in Phase 2 Round 3. Therefore, $\mathcal{A}$ is not able to impersonate $\mathbf{C}$ in the Join Protocol.

We consider case (b) as follows: as the adversary $\mathcal{A}$ is not a joining user, therefore, $\mathcal{A}$ can only obtain the public exchange information sent in the Join Protocol. First, we consider if $\mathcal{A}$ is able to obtain the new subgroup key K'_J from the public information sent. But by ignoring the authentication information $(\mu'_{J,k}, \bar{\mu}'_{J,k}, \hat{\mu}'_{J,k}$ and $\theta'_J)$ sent, the proposed Join protocol to form the new subgroup key K'_J is just the **ESTAR** protocol executed by $\mathbf{S}_J$ and $U_{J,k}$ for $n_J + 1 \leq k \leq n_J + m$, where Phase 1 Round 2, Phase 1 Round 3 together with Phase 3 Round 1, and Phase 3 Round 2 of

the Join Protocol is just the Round 1, Round 2 and Computation of Key of the ESTAR protocol respectively. By Theorem 41, it is shown that the ESTAR protocol is secure. Hence, $\mathcal{A}$ is not able to obtain the subgroup key K'_J in this case. The new subgroup key K'_J is also the plaintext of $\hat{c}_J'' = \mathcal{E}_{K_J}(K'_J || K' || S_J || \text{SID}'_J || \tau'_J)$. As $\mathcal{A}$ does not know the existing subgroup key K_J and the symmetric key encryption scheme SKE is secure, therefore $\mathcal{A}$ is not able to obtain new subgroup key K'_J from $\hat{c}_J''$.

Next, we consider if $\mathcal{A}$ is able to obtain the new group key K' from the public information sent in the Join Protocol. But by ignoring the authentication information $(\mu'_J, \bar{\mu}'_J, \hat{\mu}'_J, \hat{\mu}'_i$ and $\theta'_C)$ sent, the proposed Join protocol to form the new final group key K' is just the ESTAR protocol, where Phase 2 Round 2 (S_J sends new $(y_J^*)'$ while S_i for $1 \leq i \neq J \leq n_c$ uses y_i^* that C knows from previous GKA), Phase 2 Round 3 and Phase 3 Round 1 of the Join Protocol is just Round 1, Round 2 and Computation of Key of the ESTAR protocol respectively. By Theorem 41, it is shown that the ESTAR protocol is secure. Hence, $\mathcal{A}$ is not able to obtain the new final group key K' in this case. The new final group key K' is the plaintext of $\hat{c}_J'' = \mathcal{E}_{K_J}(K'_J || K' || S_J || \text{SID}'_J || \tau'_J)$ and $\hat{c}_i'' = \mathcal{E}_{K_i}(K' || S_i || \text{SID}'_i || \tau'_i)$ for $1 \leq i \neq J \leq n_c$. As $\mathcal{A}$ does not know the existing subgroup key K_J and K_i for $1 \leq i \neq J \leq n_c$ and the symmetric key encryption scheme SKE is secure, therefore $\mathcal{A}$ is not able to obtain new final group key K' from $\hat{c}_J''$ and $\hat{c}_i''$ for $1 \leq i \neq J \leq n_c$. The new final group key K' is also the plaintext of $\hat{c}'_{J,k} = \mathcal{E}_{t'_{J,k}}(K'_J || K' || S_J || z'_{J,k} || \text{SID}'_J)$. As $\hat{c}'_{J,k}$ is part of the information sent in the ESTAR protocol to establish the new subgroup key K'_J and both K'_J and K' are the plaintext of $\hat{c}'_{J,k}$, hence $\mathcal{A}$ is also not able to obtain final group key K' from $\hat{c}'_{J,k}$ as the ESTAR protocol is secure. Hence, the adversary $\mathcal{A}$ is not able to obtain new subgroup key K'_J and new final group key K' from the proposed password-based Join Protocol. ■

We shall now prove that an adversary is not able to launch a DoS attack against the proposed password-based Join protocol. The theorem is stated as follows:

Theorem 47 *Assume that the computation of the hash functions H_1 and H_2 , the message authentication code scheme MACS and the symmetric-key encryption scheme SKE is efficient, then the proposed password-based Join protocol is resistant to DoS attacks.*

Proof: First, the adversary can only try to launch DoS attack on subgroup controller S_J in Phase 1 as the other subgroup controllers S_i for $1 \leq i \neq J \leq n_c$ do not send and receive

information in Phase 1. The proof of this is the same as the adversary try to launch DoS attack on subgroup controller S_i in the IKA protocol as shown in Theorem 44. Hence we omit the proof here. The other way to launch a DoS attack on subgroup controller S_J and subgroup controllers S_i for $1 \leq i \neq J \leq n_c$ is by invalid main controller. The proof of this is the same as the adversary tries to launch a DoS attack on subgroup controller S_i by invalid main controller in the IKA protocol as shown in Theorem 44. Hence we omit the proof here.

Now, consider an adversary that tries to launch a DoS attack on the main controller C in the Join Protocol. As only subgroup controller S_J sends information to main controller C in Round 2 of Phase 2 while other subgroup controllers S_i for $1 \leq i \neq J \leq n_c$ do not send any information in Phase 2 of the Join Protocol, hence main controller C will reject all invalid messages sent by the adversary and only accept the valid message sent by the subgroup controller S_J . If the adversary tries to launch DoS attack by sending μ'_J to the main controller in Round 2 of Phase 2, then the main controller C is able to detect the invalidity of μ'_J by the efficient computation of message authentication code and immediately reject the adversary as the adversary does not know the encryption key ek_J and authentication key ak_J to compute the correct μ'_J . In this case, the adversary is not able to launch DoS attack on the main controller. The adversary is also not able to replay old messages in Round 1 and Round 2 of Phase 2 because the τ'_c , $\bar{\tau}'_j$ and $\bar{\tau}'_i$ for $1 \leq i \neq J \leq n_c$ and SID'_c sent by C in Round 1 of Phase 2 are fresh for every protocol run. ■

Security Analysis of Leave Protocol

Theorem 48 *The proposed Leave protocol achieves forward secrecy provided the ESTAR protocol and the symmetric key encryption scheme SKE are secure.*

Proof: First, consider a leaving user, say U_{L,l_k} ($1 \leq k \leq \ell$) tries to obtain the new subgroup key $K'_L = g^{r'_L r'_L + \sum_{j=1}^{n_L - \ell} r'_L r_{L,j}}$ (where $r_{L,j}$ belongs to remaining users $V_{L,j}$ for $1 \leq j \leq n_L - \ell$) from the previous subgroup key $K_L = g^{r_L r_L + \sum_{j=1}^{n_L} r_L r_{L,j}}$. As r'_L and r_L are random and independent to each other, therefore $g^{r'_L r'_L}$ and $g^{r_L r_L}$ are independent to each other and $g^{r'_L r_{L,j}}$ and $g^{r_L r_{L,j}}$ for $1 \leq j \leq n_L - \ell$ are also independent to each other. Furthermore, there are $n_L - \ell$ terms of $g^{r'_L r_{L,j}}$ in new subgroup key K'_L while there are n_L terms of $g^{r_L r_{L,j}}$ in previous subgroup key K_L , therefore K'_L and K_L are independent to each other. Hence, leaving user U_{L,l_k} is not able to obtain

K'_L from K_L . Then U_{L,l_k} only has the public information sent in the Leave Protocol. But by ignoring the authentication information ($\hat{\mu}'_{L,j}$ and θ'_L) sent, the proposed Leave protocol to form the new subgroup key K'_L is just the ESTAR protocol executed by S_L and remaining users $V_{L,j}$ for $1 \leq j \leq n_L - \ell$, where Phase 1 Round 1 together with Phase 3 Round 1, and Phase 3 Round 2 is just Round 2 and Computation of Key of the ESTAR protocol respectively (we omit Round 1 of the ESTAR protocol as $V_{L,j}$ for $1 \leq j \leq n_L - \ell$ uses $z^*_{L,j}$ that S_L already knows from previous GKA). By Theorem 41, it is shown that the ESTAR protocol is secure. Hence, leaving user U_{L,l_k} ($1 \leq k \leq \ell$) is not able to obtain the new subgroup key K'_L from the Leave Protocol.

Next we consider the leaving user U_{L,l_k} ($1 \leq k \leq \ell$) tries to obtain the new final group key $K' = g^{r'_c r'_c + r'_c H_2(K'_L) + \sum_{i=1, i \neq L}^{n_c} r'_c H_2(K_i)}$ from the previous final group key $K = g^{r_c r_c + \sum_{i=1}^{n_c} r_c H_2(K_i)}$. As r'_c and r_c are random and independent to each other, therefore, $g^{r'_c r'_c}$ and $g^{r_c r_c}$ are independent to each other and $g^{r'_c H_2(K_i)}$ and $g^{r_c H_2(K_i)}$ for $1 \leq i \neq L \leq n_c$ are independent to each other. Furthermore, as K'_L and K_L are random and independent to each other, therefore $H_2(K'_L)$ and $H_2(K_L)$ are independent to each other as the output of the hash function H_2 is independent. Hence, $g^{r'_c H_2(K'_L)}$ and $g^{r_c H_2(K_L)}$ are also independent to each other. Hence K' and K are independent to each other and leaving user U_{L,l_k} is not able to obtain K' from K . Therefore, U_{L,l_k} only has the public information sent in the Leave Protocol. But by ignoring the authentication information ($\mu'_L, \bar{\mu}'_L, \hat{\mu}'_L, \hat{\mu}'_i$ and θ'_c) sent, the proposed Leave protocol to form the new final group key K' is just the ESTAR protocol, where Phase 2 Round 2 (S_L sends new $(y_L^*)'$ while S_i for $1 \leq i \neq L \leq n_c$ uses y_i^* that C knows from previous GKA), Phase 2 Round 3 and Phase 3 Round 1 is just Round 1, Round 2 and Computation of Key of the ESTAR protocol. By Theorem 41, it is shown that the ESTAR protocol is secure. Hence, leaving user U_{L,l_k} ($1 \leq k \leq \ell$) is not able to obtain the new final group key K' in this case. The new final group key K' is also the plaintext of $\hat{c}'_{L,j} = \mathcal{E}_{t'_{L,j}}(K'_L || K' || S_L || z_{L,j} || \text{SID}'_L)$ and $\hat{c}''_i = \mathcal{E}_{K_i}(K' || S_i || \text{SID}'_i || \tau'_i)$ for $1 \leq i \neq L \leq n_c$ respectively. As $\hat{c}'_{L,j}$ is part of the information sent in the ESTAR protocol to establish the new subgroup key K'_L and both K'_L and K' are the plaintext of $\hat{c}'_{L,j}$, hence leaving user U_{L,l_k} ($1 \leq k \leq \ell$) is not able to obtain final group key K' from $\hat{c}'_{L,j}$ as the ESTAR protocol is secure. As leaving user U_{L,l_k} ($1 \leq k \leq \ell$) does not know the existing subgroup key K_i for $1 \leq i \neq L \leq n_c$ and the symmetric key encryption scheme SKE is secure, therefore leaving user U_{L,l_k} ($1 \leq k \leq \ell$) is not able to obtain new final group key K' from $\hat{c}''_i$ for

$1 \leq i \neq L \leq n_c$. Hence, the Leave Protocol achieves backward secrecy. $\blacksquare$

The Leave protocol also has to ensure that the adversary $\mathcal{A}$ (not a leaving user $\mathbf{U}_{L,l_k}$ ($1 \leq k \leq \ell$)) who eavesdrops on the communication cannot obtain the new subgroup key K'_L and new final group key K' from the Leave Protocol.

Theorem 49 *An adversary $\mathcal{A}$ (not a leaving user $\mathbf{U}_{L,l_k}$ ($1 \leq k \leq \ell$)) cannot obtain the new subgroup key K'_L and new final group key K' in the proposed password-based Leave Protocol provided the ESTAR protocol, the symmetric key encryption scheme SKE and the message authentication code scheme MACS are secure.*

Proof: In this proof, we consider two cases, that is (a) the adversary $\mathcal{A}$ tries to impersonate subgroup controller $\mathbf{S}_L$, subgroup controller $\mathbf{S}_i$ for $1 \leq i \neq L \leq n_c$ or main controller $\mathbf{C}$ to establish the new subgroup key K'_L and new final group key K' (we do not consider $\mathcal{A}$ tries to impersonate remaining users $\mathbf{V}_{L,j} \in \mathcal{G}'_L$ for $2 \leq j \leq n_L - \ell$ and users $\mathbf{U}_{i,j}$ for $1 \leq i \neq L \leq n_c$ and $2 \leq j \leq n_i$ as $\mathbf{V}_{L,j}$ and $\mathbf{U}_{i,j}$ do not send any information in the Leave Protocol) and (b) the adversary $\mathcal{A}$ tries to obtain the new subgroup key K'_L and new final group key K' from the public information in the Leave Protocol.

For case (a), we first consider $\mathcal{A}$ tries to impersonate subgroup controller $\mathbf{S}_L$. As $\mathbf{S}_L$ does not send any information in Phase 1, hence we first consider $\mathcal{A}$ tries to impersonate $\mathbf{S}_L$ in Phase 2. The proof that $\mathcal{A}$ is not able to impersonate $\mathbf{S}_L$ in Phase 2 of the Leave Protocol is the same as the proof that $\mathcal{A}$ is not able to impersonate $\mathbf{S}_i$ in Phase 2 of the IKA protocol as shown in Theorem 43. Hence we omit the proof here. Next, consider $\mathcal{A}$ impersonating $\mathbf{S}_L$ in Phase 3 Round 1. Then $\mathcal{A}$ selects random $\tau'_L \in Z_q^*$, $\tau'_{L,j} \in Z_q^*$ for $1 \leq j \leq n_L - \ell$ and session identity SID'_L . Next, $\mathcal{A}$ needs to compute valid $c'_{L,j} = \mathcal{E}_{ek_{L,j}}(\tau'_L || \tau'_{L,j} || \mathbf{S}_L || \mathbf{V}_{L,j} || \text{SID}'_L)$, $v'_{L,j} = \text{H}_1(\text{pw}_{L,j} || \tau'_L || \tau'_{L,j} || \text{SID}'_L)$, $\hat{c}'_{L,j} = \mathcal{E}_{t'_{L,j}}(K'_L || K' || \mathbf{S}_L || z_{L,j} || \text{SID}'_L)$, $\hat{\mu}'_{L,j} = \text{MAC}_{ak_{L,j}}(\tau'_L || \tau'_{L,j} || z'_L || \theta'_L || \text{SID}'_L || \mathbf{S}_L || \mathbf{V}_{L,j} || 11)$ and $\theta'_L = \text{H}_1(\text{SID}'_L || \text{UID}'_L || K'_L || K' || \tau'_L || z'_L)$. The validity of $c'_{L,j}$, $v'_{L,j}$, $\hat{c}'_{L,j}$ and $\hat{\mu}'_{L,j}$ are checked by $\mathbf{V}_{L,j}$ for $1 \leq j \leq n_L - \ell$ in Phase 3 Round 2. If either one of $c'_{L,j}$, $v'_{L,j}$, $\hat{c}'_{L,j}$ or $\hat{\mu}'_{L,j}$ is not correct, then $\mathbf{V}_{L,j}$ rejects invalid $\mathbf{S}_L$. As $\mathcal{A}$ does not know the encryption key $ek_{L,j}$ of $\mathbf{V}_{L,j}$ and the symmetric key encryption scheme SKE is secure, therefore $\mathcal{A}$ is not able to produce a valid $c'_{L,j}$. $\mathcal{A}$ also does not know password $\text{pw}_{L,j}$ of $\mathbf{V}_{L,j}$, hence $\mathcal{A}$ is not able to produce a valid $v'_{L,j}$. As $\mathcal{A}$ also does not know the authentication key $ak_{L,j}$ of $\mathbf{V}_{L,j}$ and the message authentication code scheme MACS is

assumed to be secure, therefore $\mathcal{A}$ is not able to compute a valid $\hat{\mu}'_{L,j}$. Furthermore, as mentioned earlier, $\mathcal{A}$ does not know password $\text{pw}_{L,j}$ of $V_{L,j}$, hence $\mathcal{A}$ is not able to obtain $z_{L,j}$ from $z_{L,j}^*$ to compute valid $t'_{L,j} = g^{r_L r_{L,j}}$. Therefore, $\mathcal{A}$ is not able to compute the valid ciphertext $\hat{c}'_{L,j}$ as the symmetric key encryption scheme SKE is secure. Hence $\mathcal{A}$ is not able to impersonate S_L in Phase 3. Therefore, $\mathcal{A}$ is not able to impersonate S_L in the Leave Protocol.

Next, we consider $\mathcal{A}$ wishes to impersonate subgroup controller S_i for $1 \leq i \neq L \leq n_c$ in the Leave Protocol. As S_i for $1 \leq i \neq L \leq n_c$ does not send any information in Phase 1 and 2, hence we only consider $\mathcal{A}$ tries to impersonate S_i in Phase 3. Then in Phase 3 Round 1, $\mathcal{A}$ has to generate valid $\hat{c}_i'' = \mathcal{E}_{K_i}(K' || S_i || \text{SID}'_i || \tau'_i)$ and $\theta'_i = H_1(\text{SID}'_i || \text{UID}_i || K' || \tau'_i || z_i)$. The validity of $\hat{c}_i''$ is checked by $U_{i,j}$ for $1 \leq i \neq L \leq n_c$ and $1 \leq j \leq n_i$ in Phase 3 Round 2. If $\hat{c}_i''$ is not correct, then $U_{i,j}$ rejects invalid S_i . As $\mathcal{A}$ does not know the subgroup key K_i and the symmetric key encryption scheme SKE is secure, therefore $\mathcal{A}$ is not able to compute the valid ciphertext $\hat{c}_i''$. Hence, $\mathcal{A}$ is not able to impersonate S_i in the Leave Protocol.

Next, we consider $\mathcal{A}$ tries to impersonate C . Then in Phase 2 Round 1, $\mathcal{A}$ selects randoms $\tau'_C \in Z_q^*$, $\bar{\tau}'_L \in Z_q^*$, $\bar{\tau}'_i \in Z_q^*$ for $1 \leq i \neq L \leq n_c$ and session identity SID'_C . Next, $\mathcal{A}$ has to compute the valid $c'_L = \mathcal{E}_{ek_L}(\tau'_C || \bar{\tau}'_L || C || S_L || \text{SID}'_C)$, $v'_L = H_1(\text{pw}_L || \tau'_C || \bar{\tau}'_L || \text{SID}'_C)$, $c'_i = \mathcal{E}_{ek_i}(\tau'_C || \bar{\tau}'_i || C || S_i || \text{SID}'_C)$ and $v'_i = H_1(\text{pw}_i || \tau'_C || \bar{\tau}'_i || \text{SID}'_C)$ for $1 \leq i \neq L \leq n_c$. The validity of c'_L and v'_L (c'_i and v'_i) are checked by S_L (S_i) in Phase 2 Round 2. If either c'_L or v'_L (c'_i or v'_i) is not valid, then S_L (S_i) rejects the invalid C . As $\mathcal{A}$ does not know the encryption key ek_L and ek_i of S_L and S_i for $1 \leq i \neq L \leq n_c$ respectively and the symmetric-key encryption scheme SKE is assumed to be secure, therefore $\mathcal{A}$ is not able to produce valid c'_L and c'_i for $1 \leq i \neq L \leq n_c$ respectively. $\mathcal{A}$ also does not know the password pw_L and pw_i of S_L and S_i for $1 \leq i \neq L \leq n_c$ respectively, therefore $\mathcal{A}$ is not able to produce valid v'_L and v'_i for $1 \leq i \neq L \leq n_c$ respectively. Hence $\mathcal{A}$ is not able to impersonate C in Phase 2 Round 1. Next consider $\mathcal{A}$ impersonating main controller C in Phase 2 Round 3. Then $\mathcal{A}$ needs to compute valid $\hat{c}'_L = \mathcal{E}_{t'_L}(K' || C || y'_L || \text{SID}'_C)$, $\hat{\mu}'_L = \text{MAC}_{ak_L}(\tau'_C || \bar{\tau}'_L || y'_C || \theta'_C || \text{SID}'_C || C || S_L || 11)$, $\hat{c}'_i = \mathcal{E}_{t'_i}(K' || C || y'_i || \text{SID}'_C)$, $\hat{\mu}'_i = \text{MAC}_{ak_i}(\tau'_C || \bar{\tau}'_i || y'_C || \theta'_C || \text{SID}'_C || C || S_i || 11)$ for $1 \leq i \neq L \leq n_c$ and $\theta'_C = H_1(\text{SID}'_C || \text{UID}_C || K' || \tau'_C || y'_C)$. The validity of $\hat{c}'_L$ and $\hat{\mu}'_L$ ($\hat{c}'_i$ and $\hat{\mu}'_i$) are checked by S_L (S_i) in Phase 3 Round 1. If either one of $\hat{c}'_L$ or $\hat{\mu}'_L$ ($\hat{c}'_i$ or $\hat{\mu}'_i$) is not correct, then S_J (S_i) rejects invalid C . As $\mathcal{A}$ does not know password pw_L and pw_i of S_L and S_i for $1 \leq i \neq L \leq n_c$ respectively, hence $\mathcal{A}$ is not able to obtain y'_L from $(y_L^*)'$

and y_i from y_i^* to compute valid $t'_L = g^{r'_c H_2(K'_L)}$ and $t'_i = g^{r'_c H_2(K_i)}$ respectively. Therefore, $\mathcal{A}$ is not able to compute the valid ciphertext $\hat{c}'_L$ and $\hat{c}'_i$ for $1 \leq i \neq L \leq n_c$ as the symmetric-key encryption scheme SKE is assumed to be secure. Furthermore, $\mathcal{A}$ does not know the authentication key ak_L and ak_i of $\mathbf{S}_L$ and $\mathbf{S}_i$ for $1 \leq i \neq L \leq n_c$ respectively and the message authentication code scheme MACS is assumed to be secure, therefore $\mathcal{A}$ is not able to compute a valid $\hat{\mu}'_L$ and $\hat{\mu}'_i$ respectively. Hence, $\mathcal{A}$ is not able to impersonate $\mathbf{C}$ in Phase 2 Round 3. Therefore, $\mathcal{A}$ is not able to impersonate $\mathbf{C}$ in the Leave Protocol.

Next we consider case (b) as follows: as the adversary $\mathcal{A}$ is not a leaving user $\mathbf{U}_{L,l_k}$ ($1 \leq k \leq \ell$), therefore, $\mathcal{A}$ can only obtain the public exchange information in the Leave Protocol. First we consider if $\mathcal{A}$ can obtain the new subgroup key K'_L from the public information sent in the Leave Protocol. But by ignoring the authentication information ($\hat{\mu}'_{L,j}$ and θ'_L) sent, the proposed Leave protocol to form the new subgroup key K'_L is just the ESTAR protocol executed by $\mathbf{S}_L$ and remaining users $\mathbf{V}_{L,j}$ for $1 \leq j \leq n_L - \ell$, where Phase 1 Round 1 together with Phase 3 Round 1, and Phase 3 Round 2 is just Round 2 and Computation of Key of the ESTAR protocol respectively (we omit Round 1 of the ESTAR protocol as $\mathbf{V}_{L,j}$ for $1 \leq j \leq n_L - \ell$ uses $z_{L,j}^*$ that $\mathbf{S}_L$ already knows from previous GKA). By Theorem 41, it is shown that the ESTAR protocol is secure. Hence, $\mathcal{A}$ is not able to obtain the new subgroup key K'_L from the Leave Protocol.

Next, we consider if $\mathcal{A}$ is able to obtain the new group key K' from the public information sent in the Leave Protocol. But by ignoring the authentication information (μ'_L , $\bar{\mu}'_L$, $\hat{\mu}'_L$, $\hat{\mu}'_i$ and θ'_c) sent, the proposed Leave protocol to form the new final group key K' is just the ESTAR protocol, where Phase 2 Round 2 ($\mathbf{S}_L$ sends new $(y_L^*)'$ while $\mathbf{S}_i$ for $1 \leq i \neq L \leq n_c$ uses y_i^* that $\mathbf{C}$ knows from previous GKA), Phase 2 Round 3 and Phase 3 Round 1 is just Round 1, Round 2 and Computation of Key of the ESTAR protocol. By Theorem 41, it is shown that the ESTAR protocol is secure. Hence, $\mathcal{A}$ is not able to obtain the new final group key K' in this case. The new final group key K' is also the plaintext of $\hat{c}'_{L,j} = \mathcal{E}_{t'_{L,j}}(K'_L || K' || \mathbf{S}_L || z_{L,j} || \text{SID}'_L)$ and $\hat{c}''_i = \mathcal{E}_{K_i}(K' || \mathbf{S}_i || \text{SID}'_i || \tau'_i)$ for $1 \leq i \neq L \leq n_c$ respectively. As $\hat{c}'_{L,j}$ is part of the information sent in the ESTAR protocol to establish the new subgroup key K'_L and both K'_L and K' are the plaintext of $\hat{c}'_{L,j}$, hence $\mathcal{A}$ is not able to obtain final group key K' from $\hat{c}'_{L,j}$ as the ESTAR protocol is secure. As $\mathcal{A}$ does not know the existing subgroup key K_i for $1 \leq i \neq L \leq n_c$ and the symmetric key encryption scheme SKE is

secure, therefore $\mathcal{A}$ is not able to obtain new final group key K' from $\hat{c}_i''$ for $1 \leq i \neq L \leq n_c$. Hence, the adversary $\mathcal{A}$ is not able to obtain new subgroup key K'_L and new final group key K' from the proposed password-based Leave Protocol. ■

Lastly, we prove that an adversary is not able to launch a DoS attack against the proposed password-based Leave protocol. The theorem is stated as follows:

Theorem 50 *Assume that the computation of the hash functions H_1 and H_2 , the message authentication code scheme MACS and the symmetric-key encryption scheme SKE is efficient, then the proposed password-based Leave protocol is resistant to DoS attacks.*

Proof: The adversary is not able to launch a DoS attack on subgroup controller S_L and S_i for $1 \leq i \neq L \leq n_c$ in Phase 1 of the Leave Protocol as no information is sent in Phase 1. The adversary can only try to launch DoS attack on subgroup controller S_L and subgroup controllers S_i for $1 \leq i \neq L \leq n_c$ in Phase 2 by invalid main controller. The proof of this is the same as the adversary tries to launch a DoS attack on subgroup controller S_i by invalid main controller in the IKA protocol as shown in Theorem 44. Hence we omit the proof here.

Now, consider an adversary that tries to launch a DoS attack on the main controller C in the Leave Protocol. As only subgroup controller S_L sends information to main controller C in Round 2 of Phase 2 while other subgroup controllers S_i for $1 \leq i \neq L \leq n_c$ do not send any information in Phase 2 of the Leave Protocol, hence main controller C will reject all invalid messages sent by the adversary and only accept the valid message sent by the subgroup controller S_L . If the adversary tries to launch DoS attack by sending μ'_L to the main controller in Round 2 of Phase 2, then the main controller C is able to detect the invalidity of μ'_L by the efficient computation of message authentication code and immediately reject the adversary as the adversary does not know the encryption key ek_L and authentication key ak_L to compute the correct μ'_L . In this case, the adversary is not able to launch DoS attack on the main controller. The adversary is also not able to replay old messages in Round 1 and Round 2 of Phase 2 because the τ'_C , $\bar{\tau}'_L$ and $\bar{\tau}'_i$ for $1 \leq i \neq L \leq n_c$ and SID'_C sent by C in Round 1 of Phase 2 are fresh for every protocol run. ■

10.3.2 Complexity Analysis

In Table 10.2, we first analyze and compare the computational (number of modular exponentiations performed) and communication (number of messages sent and received) costs of three password-based Initial Group Key Agreement (IKA) protocols, namely Abdalla et al's protocol [ABCP06], Byun et al's EKE-D protocol [BLLH06] and our proposed password-based IKA protocol. We selected Abdalla et al's protocol [ABCP06] for comparison as it is an efficient password-based IKA protocol that requires only five constant rounds regardless of the number of user nodes while Byun et al's EKE-D protocol [BLLH06] was selected for comparison as it is also an efficient password-based IKA protocol that has a controller to help reduce the computation workload of the low-power user nodes. It should be noted that Abdalla et al's protocol [ABCP06] uses a single common password amongst all users while Byun et al's EKE-D protocol [BLLH06] and our proposed password-based GKA protocol uses different passwords shared between the server/controller and each user node. Next, we analyze and compare the computational and communication costs of dynamic protocols, namely Join and Leave Protocols. As Abdalla et al did not include any dynamic (Join and Leave) protocols in their paper [ABCP06], hence we only compare the dynamic (Join and Leave) protocols of Byun et al's EKE-D protocol [BLLH06] and our proposed dynamic (Join and Leave) Protocols.

Initial Group Key Agreement (IKA) Protocol

From Table 10.2, it can be seen that user nodes $U_{i,j}$ for $1 \leq i \leq n_c$ and $1 \leq j \leq n_i$ in the proposed password-based IKA protocol only has to perform 3 modular exponentiations, 1 message transmission and 2 message receptions. Although user nodes in EKE-D IKA protocol performs only 2 modular exponentiations, however each user in EKE-D IKA protocol has to send 2 messages and receive $n + 1$ messages, which is rather inefficient. The users in Abdalla et al's protocol has to receive $4(n - 1)$ messages, which is the most number of messages to be received. The controller node in EKE-D IKA protocol performs the most $(2n)$ number of modular exponentiations as compared to $2n_c + 2$ and $2n_i + 5$ modular exponentiations by main controller C and subgroup controllers S_i for $1 \leq i \leq n_c$ in the proposed password-based IKA protocol respectively. Although controller node in EKE-D IKA protocol transmits only 2 messages, the controller node in EKE-D IKA protocol has to receive $2n$ messages. In comparison, C and S_i in the proposed password-based GKA protocol

Table 10.2: Computational and Communication Cost

			Mod. Exp.	Msg Sent	Msg Rx
I K A	Abdalla et al's [ABCP06] User		3	4	$4(n-1)$
	EKE-D [BLLH06]	Controller node	$2n$	2	$2n$
		User nodes	2	2	$n+1$
	Proposed IKA Protocol	C	$2n_c+2$	2	n_c
		S_i	$2n_i+5$	3	n_i+2
		$U_{i,j}$	3	1	2
J O I N	EKE-D [BLLH06]	Controller node	$n+2m$	2	$n+2m$
		Existing user nodes	0	2	$m+1$
		New joining users	2	2	$n+2$
	Proposed Join Protocol	C	n_c+3	2	1
		S_J	$2m+5$	3	$m+2$
		$S_i \neq S_J$	1	1	2
		$U_{J,k} \in \mathcal{J}$	3	1	2
		$U_{J,j} \in \mathcal{G}_J$	0	0	1
		$U_{i,j}$ for $i \neq J$	0	0	1
L E A V E	EKE-D [BLLH06]	Controller node	$2(n-\ell)$	2	$2(n-\ell)$
		Remaining user nodes	2	2	$n-\ell+1$
	Proposed Leave Protocol	C	n_c+3	2	1
		S_L	$n_L-\ell+5$	2	2
		$S_i \neq S_L$	1	1	2
		$V_{L,j}$	1	0	1
		$U_{i,j}$ for $i \neq L$	0	0	1

only transmit 2 and 3 messages and receive n_c and n_i+2 messages respectively.

Join Protocol

Although the new joining users in EKE-D Join protocol only has to compute 2 modular exponentiations as compared to 3 modular exponentiations by new users $U_{J,k}$ for $n_J+1 \leq k \leq n_J+m$ in our proposed Join Protocol, however the joining users in EKE-D Join protocol has to receive $n+2$ messages, where n is the entire group size. In comparison, the new users $U_{J,k}$ in our proposed Join Protocol only has to receive 2 messages. All other users $U_{J,j}$ for $1 \leq j \leq n_J$ and $U_{i,j}$ for $1 \leq i \neq J \leq n_c$ and $1 \leq j \leq n_i$ in our proposed Join Protocol only has to receive 1 message while other users in EKE-D has to send 2 messages and receive $m+1$ messages. The controller node in EKE-D Join Protocol has to compute the most number of exponentiations ($n+2m$ exponentiations)

and receive the most number of messages ($n + 2m$ messages). In comparison, the main controller C , subgroup controller S_J and subgroup controller S_i for $1 \leq i \neq J \leq n_c$ only has to perform $n_c + 3$, $2m + 5$ and 1 modular exponentiations respectively and receive 1, $m + 2$ and 2 messages respectively in our proposed Join Protocol.

Leave Protocol

The users $U_{i,j}$ for $1 \leq i \neq L \leq n_c$ and $1 \leq j \leq n_i$ in our proposed Leave Protocol only has to receive 1 message while the remaining users $V_{L,j}$ for $1 \leq j \leq n_L - \ell$ in our proposed Leave Protocol only has to perform 1 exponentiations and receive 1 message. In comparison, the remaining users in EKE-D Leave protocol has to perform 2 exponentiations, send 2 messages and receive $n - \ell + 1$ messages. The controller node in EKE-D Leave Protocol has to perform the most number of exponentiations ($2(n - \ell)$ exponentiations) and receive the most number of messages ($2(n - \ell)$ messages). In comparison, main controller C , subgroup controller S_L and subgroup controller S_i for $1 \leq i \neq L \leq n_c$ in our proposed Leave Protocol only has to perform $n_c + 3$, $n_L - \ell + 5$ and 1 exponentiations each respectively and receive 1, 2 and 2 messages respectively. All other subgroup controllers S_i for $1 \leq i \neq L \leq n_c$ only has to send and receive 1 message in our proposed Leave Protocol.

10.3.3 Energy Analysis

In this section, we perform the computational and communication energy consumption analysis. Our computational energy costs is based on the 206MHz “StrongARM” microprocessor as discussed in Section 3.3.2 while our communication energy costs is based on the Lucent WaveLAN card discussed in Section 3.4.2. In Section 10.3.3, we provide a chart which compares the energy costs of users in our proposed password-based Initial GKA (IKA) protocol and two password-based GKA protocols, namely Abdalla et al’s protocol [ABCP06] and Byun et al’s EKE-D protocol [BLLH06], for different number of user nodes. We also provide the energy cost comparison of user nodes running the proposed Join and Leave Protocol and EKE-D Join and Leave Protocol respectively.

Initial Group Key Agreement (IKA) Protocol

Based on the complexity analysis in Table 10.2 and the energy costs in Tables 3.3 and 3.5 in Chapter 3, we obtain the total energy costs of user nodes running Abdalla et al's protocol [ABCP06], Byun et al's EKE-D protocol [BLLH06] and our proposed password-based Initial GKA (IKA) protocol respectively in Figure 10.1. We do not show the energy costs for the main controller C and subgroup controllers S_i in our proposed password-based IKA protocol and the controller node in EKE-D as they do not have energy constraints. The Y-axis in the figure measures the energy consumed in milliJoules (mJ) while the X-axis shows the total number of user nodes (1000, 2500, 7500 and 10000 respectively). For our proposed password-based IKA protocol, the number of subgroups $n_c = 10$ while the number of user nodes in each subgroup (excluding subgroup controller S_i) are 100, 250, 750 and 1000 for total number of user nodes 1000, 2500, 7500 and 10000 respectively. The numbers shown inside the boxes in Figure 10.1 shows the total energy costs (in mJ) for each user node in Abdalla et al's protocol [ABCP06], Byun et al's EKE-D protocol [BLLH06] and our proposed password-based IKA protocol respectively. For example, when the number of user nodes n_i is 7500, the users $U_{i,j}$ for $1 \leq i \leq n_c$ and $1 \leq j \leq n_i$ in the proposed password-based IKA protocol consumes only 131.59 mJ as compared to 980.59 mJ by the user nodes in EKE-D and 2139.24 mJ consumed by user nodes in Abdalla et al's protocol.

Figure 10.1: Energy Costs of user nodes for Initial Group Key Agreement (IKA) Protocol.

Although the proposed password-based IKA protocol provides protection against DoS attacks

and is secure against both off-line and on-line password guessing attacks, it can be seen from Figure 10.1 that the user nodes $U_{i,j}$ for $1 \leq i \leq n_c$ and $1 \leq j \leq n_i$ in the proposed password-based IKA protocol are the most energy efficient for different number of user nodes (1000, 2500, 7500 and 10000). Furthermore, the proposed password-based IKA protocol is also the most scalable due to the use of subgroups in the Central-Central group model which helps to reduce the number of computations and messages received by users $U_{i,j}$ despite the large group size. User nodes in Abdalla et al's protocol have the highest energy costs mainly due to the large number of messages to be received from other nodes. Similarly, the main factor for high energy costs of user nodes in EKE-D protocol is also due to the large amount of messages ($\mathcal{O}(n)$) received by each user node.

Join Protocol

Table 10.3 shows the total energy costs of different user nodes when running the EKE-D Join protocol and the proposed Join Protocol. These results are based on the complexity analysis in Table 10.2 and the energy costs described in Tables 3.3 and 3.5 in Chapter 3. We assume that existing group size $n = 7500$ and there are $m = 20$ new users joining the group. The number of user nodes in each subgroup $\mathcal{G}_i$ in our proposed Join Protocol is $n_i = 750$ since group size $n = 7500$.

Table 10.3: User Energy Costs for Join Protocol(20 Users)

		Energy Consumed (mJ)
EKE-D [BLLH06]	Existing users	80.94
	New joining users	731.95
Proposed Join Protocol	$U_{J,k} \in \mathcal{J}$	50.34
	$U_{J,j} \in \mathcal{G}_J$	3.20
	$U_{i,j}$ for $i \neq J$	0.89

As shown in Table 10.3, our proposed Join Protocol is the most efficient, with the joining users $U_{J,k}$ for $n_J + 1 \leq k \leq n_J + m$ consuming $50.34mJ$ while other users $U_{J,j}$ for $1 \leq j \leq n_J$ and $U_{i,j}$ for $1 \leq i \neq J \leq n_c$ and $1 \leq j \leq n_i$ consuming only $3.2mJ$ and $0.89mJ$ respectively. In comparison, the joining users in EKE-D Join Protocol consumes the largest energy ($731.95mJ$) each, which is more than 20 times higher than the joining users $U_{J,k}$ in our proposed Join Protocol. Existing users in EKE-D Join Protocol consumes $80.94mJ$, which is also very high as compared to other users $U_{J,j}$ for $1 \leq j \leq n_J$ and $U_{i,j}$ for $1 \leq i \neq J \leq n_c$ and $1 \leq j \leq n_i$ in our proposed Join Protocol.

Leave Protocol

Table 10.4 shows the total energy costs of different user nodes when running the EKE-D Leave protocol and the proposed Leave Protocol. These results are based on the complexity analysis in Table 10.2 and the energy costs described in Tables 3.3 and 3.5 in Chapter 3. Again, we assume that the existing group size $n = 7500$ and there are $\ell = 20$ users leaving the group. It should be noted that the number of user nodes in each subgroup $\mathcal{G}_i$ in our proposed Leave Protocol is $n_i = 750$ since group size $n = 7500$.

Table 10.4: User Energy Costs for Leave Protocol (20 Users)

		Energy Consumed (mJ)
EKE-D [BLLH06]	Remaining users	978.03
Proposed Leave Protocol	$V_{L,j}$	108.72
	$U_{i,j}$ for $i \neq L$	0.89

From Table 10.4, it can be seen that our proposed Leave Protocol is more efficient than EKE-D Leave protocol. The remaining users $V_{L,j}$ for $1 \leq j \leq n_L - \ell$ in our proposed Leave Protocol only has to consume $108.72mJ$, which is about 9 times lower than the energy consumed ($978.03mJ$) by remaining users in EKE-D Leave Protocol. The other users $U_{i,j}$ for $1 \leq i \neq L \leq n_c$ and $1 \leq j \leq n_i$ in our proposed Leave Protocol is the most efficient, consuming only $0.89mJ$ for a Leave event.

10.4 Summary

In the first part of this chapter, we showed that Wan et al's Authentication and Key Exchange Protocol does not provide protection against DoS attacks. We then propose a new authentication and key exchange protocol that truly provides DoS resistance and user anonymity. A security and complexity analysis showed that our proposed protocol is both secure and efficient for authentication and key exchange in wireless networks. In the second part of this chapter, we proposed authenticated password-based GKA protocols for wireless networks that provides protection against DoS attacks and is secure against both off-line and on-line password guessing attacks. Besides providing key confirmation properties, the proposed password-based GKA protocols are also energy efficient and scalable.

Chapter 11

Conclusion and Recommendations

As the last chapter of this thesis, a summary of the thesis and a discussion of open problems and possible future research directions are presented.

11.1 Conclusion

In this thesis, we have achieved all the four research goals identified in Section 1.5 as explained below:

1. To study on current GKA protocols in order to gain better understanding and improve on efficiency in terms of authentication, computation and communications aspects.
 - In Chapter 4, we studied a popular and efficient GKA protocol, the BD protocol, that is suitable for GKA in wireless network and identified an interesting characteristic that could be used to design efficient authenticated GKA protocols based on the BD. We designed an authenticated GKA protocol using this characteristic and compared it with current methods used to authenticate the BD protocol and showed that the proposed protocol is more efficient.
 - In Chapter 6, we explored ID-based authentication, which is an efficient alternative authentication techniques to using digital signatures, and proposed ID-based authenticated GKA protocols (Initial GKA, Join, Leave, Merge and Partition Protocols). Besides com-

plexity analysis, we also provide an energy analysis of the proposed protocols. Both the complexity and energy analysis shows that our proposed ID-based GKA protocols are more efficient and the energy consumed by nodes running our proposed protocols are significantly lower.

- In Chapter 10, we explored password-based authentication, which is more efficient than digital signature-based authentication in GKA protocols. We proposed an authenticated Initial Password-based GKA protocol and two authenticated dynamic Password-based GKA protocols, namely the Join and Leave Protocol, that are energy-efficient, scalable and also provides protection against DoS attacks in wireless networks. The complexity and energy analysis indicates that our proposed Password-based GKA protocols are more efficient and scalable than current GKA protocols despite providing protection against DoS attacks.
2. To examine current dynamic GKA protocols, namely Join, Leave, Merge and Partition Protocol, and design more efficient dynamic GKA protocols for wireless networks.
- In Chapter 5, we studied new research ideas to provide dynamic GKA protocols for the efficient BD protocol and showed that a security flaw in a method briefly mentioned by Steiner et al to provide a Join protocol for the BD. We also show another security flaw in the provable-secure dynamic GKA protocol of Dutta and Barua [DB05]. Next, we proposed four authenticated GKA protocols namely Join, Leave, Merge and Partition protocols for MANETs that also provides autoconfiguration which is useful when used in implementations of MANETs in real-world application. The complexity analysis of our proposed protocols with four current dynamic GKA protocols show that our proposed protocols are efficient and suitable for MANETs.
 - In Chapter 6, we proposed an ID-based authenticated Initial GKA protocol and four ID-based authenticated dynamic GKA protocols, namely Join, Leave, Merge and Partition Protocols. We also provided an energy analysis besides complexity analysis of our proposed protocols that shows that our proposed protocols are more efficient and the energy consumed by nodes running our proposed protocols are significantly lower.

- In Chapter 9 and 10, we proposed dynamic GKA protocols that use different hierarchical group structures respectively. The complexity and energy analysis of our dynamic GKA protocols in both Chapters 9 and 10 with current dynamic GKA protocols showed that both our proposed GKA protocols in Chapter 9 and 10 are more energy-efficient and scalable.
3. To design GKA protocols that are scalable and energy-efficient for large-scale wireless networks.
- In Chapter 7, we developed a hierarchical group structure known as the Generalized Circular Hierarchical (C-H) Group Model and then designed a GKA protocol for the C-H group model to be used in large scale wireless ad hoc networks. The complexity and energy analysis of the proposed protocol shows that it is more scalable and energy-efficient as compared to current GKA protocols.
 - In Chapter 8, we designed a Contributory Ring-Centralized Group Model for heterogeneous wireless networks and then proposed a GKA protocol to improve the scalability and energy efficiency of nodes in heterogeneous wireless networks. The complexity and energy analysis of our proposed protocol indicates that our proposed protocol is more efficient and suitable than current GKA protocols and therefore more suitable for heterogeneous wireless networks.
 - In Chapter 9, we proposed the authenticated initial GKA protocol and dynamic GKA protocols, namely Join and Leave Protocol, that also provides protection against DoS attacks. We showed with the complexity and energy analysis that our proposed protocols are more energy-efficient and scalable for use in heterogeneous wireless networks despite providing the important feature of protection against DoS attacks.
 - In Chapter 10, we proposed an authenticated Initial Password-based GKA protocol and two authenticated dynamic Password-based GKA protocols, namely the Join and Leave Protocol, that are energy-efficient, scalable and also provides protection against DoS attacks in wireless networks.
4. To study on the problem of Denial-of-Service (DoS) attacks in wireless networks and design

GKA protocols that provide protection against DoS attacks.

- In Chapter 9, we focus on the important problem of Denial-of-Service (DoS) attacks on GKA protocols besides focusing on scalability and energy efficiency issues in heterogeneous wireless networks. We proposed the authenticated initial GKA protocol and dynamic GKA protocols, namely Join and Leave Protocol, that offers protection against DoS attacks. We showed, by complexity and energy analysis, that our proposed protocols are more energy-efficient and scalable for use in heterogeneous wireless networks despite providing the important feature of protection against DoS attacks.
- In Chapter 10, we study further on DoS attacks. First, we point out possible DoS attacks in an Anonymous Access Control (AAC) protocol that claims to be resistant to DoS attacks. We then proposed a new and improved DoS-resistant AAC protocol. Next, we continue to explore the problem of DoS attacks against GKA protocols, including password-based GKA protocols in wireless networks. We proposed an authenticated Initial Password-based GKA protocol and two authenticated dynamic Password-based GKA protocols, namely the Join and Leave Protocol, that are energy-efficient, scalable and also provides protection against DoS attacks in wireless networks. The complexity and energy analysis indicate that our proposed Password-based GKA protocols are more efficient and scalable than current GKA protocols despite providing protection against DoS attacks.

In our results, we discovered better authentication techniques for GKA protocols through detail studies and also explored other better alternatives such as ID-based and password-based authentication as compared to the less efficient digital signature authentication. We also demonstrated that careful analysis is necessary to design dynamic GKA protocols, otherwise security flaws could occur despite the dynamic GKA protocol being proven secure. We also proposed different hierarchical group structures for different types of wireless network setting to achieve scalability and energy efficiency, especially for large scale wireless networks. Finally, we showed that the problem of DoS attacks is hard to prevent but we manage to design several GKA protocols that provide protection against DoS attacks. More importantly, we believe that the GKA protocols outlined in this thesis will provide a better understanding of constructing energy-efficient, authenticated,

scalable, dynamic and DoS-resistant GKA protocols for different types of wireless networks.

11.2 Open Problems and Future Directions

Extending our work in Chapter 4. While we have presented an efficient authenticated GKA protocol for wireless networks and also proved that the proposed protocol is secure against insider impersonation, it would be interesting to extend this work to take care of even more powerful malicious insider nodes.

Extending our work in Chapter 5 and 6. The current work focus only on the complexity analysis of our proposed protocols with current GKA protocols. It would be interesting to implement our proposed protocols in both Chapter 5 and 6 on actual MANET testbeds and analyze how efficient is the autoconfiguration feature in actual MANET implementation (for Chapter 5) and also identify how accurate our results match actual implementation (for both Chapter 5 and 6).

Extending our work in Chapter 7 and 8. In Chapters 7 and 8, we designed hierarchical group structures to help improve the scalability and efficiency of GKA protocols in different types of static wireless networks such as ad hoc networks (Chapter 7) and wireless mesh networks (Chapter 8). One interesting extension to this approach would be to implement the GKA protocols in real-world situation and compare the accuracy of the theoretical results with the actual energy costs on user devices. It should be noted however that in this case, the energy cost of the actual device e.g. PDA will include energy consumed by other peripherals such as the LCD display in the PDA. Another possible future research direction is to extend the protocols in Chapters 5, 7 and 8 to handle dynamic wireless networks by designing Join and Leave Protocols. The Join and Leave Protocols can be based on (or modified from) the Join and Leave Protocols discussed in Chapters 9 and 10. Lastly, another challenging future direction would be to design methods for the detection of colluding inside attackers in the GKA protocols.

Extending our work in Chapter 9 and 10. In Chapters 9 and 10, besides the issue of scalability, we also proposed dynamic GKA protocols that uses hierarchical group structures and are DoS-resistant. Hence, it would be interesting to implement the proposed protocols in Chapter 9

and 10 in real-world situations using real user devices such as PDAs. In this way, we would be able to see how effective the proposed protocols are in protecting against DoS attacks. We may then be able to address other DoS attacks such as Distributed DoS (DDoS) attacks and also consider protection not just for controller nodes in the network, but also the legitimate user nodes in the network.

Author's Publications

- Joseph Chee Ming Teo and Chik How Tan, "Security Analysis and Improvement of an Anonymous DoS-Resistant Access Control Protocol", IEEE International Conference on Information, Communications & Signal Processing (ICICS 2007), pp. 1 - 5, December 2007.
- Joseph Chee Ming Teo and Chik How Tan, "Denial-of-Service Resilience Password-based Group Key Agreement for Wireless Networks", ACM Q2SWiNet 2007, pp. 136 - 143, October 2007.
- Joseph Chee Ming Teo, Chik How Tan, and Jim Mee Ng, "Denial-of-Service Attack Resilience Dynamic Group Key Agreement for Heterogeneous Networks", Springer Telecommunication Systems, Vol 35. No.3-4, pp 141-160, August 2007.
- Chik How Tan and Joseph Chee Ming Teo, "Protection Against Web-based Password Phishing", Third International Conference on Information Technology: Next Generations (ITNG 2007), pp. 754-759, The IEEE Computer Society Press, 2007.
- Joseph Chee Ming Teo, Chik How Tan, and Jim Mee Ng, "Authenticated Group Key Agreement against DoS in Heterogeneous Wireless Networks", IEEE Wireless Communications and Networking Conference (WCNC), pp. 3563 - 3568, March 2007.
- Joseph Chee Ming Teo, Chik How Tan, and Jim Mee Ng, "Security Analysis of Provably Secure Constant Round Dynamic Group Key Agreement", IEICE Trans. on Fundamentals, Vol E89-A, No.11, pp.3348-3350, November 2006.
- Joseph Chee Ming Teo, Chik How Tan, and Jim Mee Ng, "Low-Power Authenticated Group Key Agreement for Heterogeneous Wireless Networks", International Journal of Security and Networks (IJSN), Vol. 1 No. 3/4, pp. 226 - 236, 2006

- Joseph Chee Ming Teo and Chik How Tan, “Authenticated Dynamic Group Key Agreement for Autoconfigurable Mobile Ad Hoc Networks”, IEICE Trans. on Communications, Vol.E89-B, No.9, pp. 2480-2492, September 2006.
- Joseph Chee Ming Teo, Chik How Tan, and Jim Mee Ng, “Low-Power Group Key Agreement for Heterogeneous Wireless Networks”, International Wireless Communications and Mobile Computing Conference 2006 (IWCMC 2006), pp. 177- 182, July 2006. *One of the selected papers for publication in International Journal of Security and Networks (IJSN).*
- Chik How Tan and Joseph Chee Ming Teo, “Energy-Efficient ID-based Group Key Agreement Protocols for Wireless Networks”, The International Workshop on Security in Systems and Networks (SSN2006, in conjunction with the 20th IEEE International Parallel and Distributed Processing Symposium (IPDPS 2006), pp. 1- 8, April 2006.
- Joseph Chee Ming Teo and Chik How Tan, “Energy Efficient and Scalable Group Key Agreement for Large Ad Hoc Network”, The 2nd ACM International Workshop on Performance Evaluation of Wireless Ad Hoc, Sensor, and Ubiquitous Networks (PE-WASUN 2005), pp.114-121, 2005.
- Chik How Tan and Joseph Chee Ming Teo, “An Authenticated Group Key Agreement for Wireless Networks”, IEEE Wireless Communications and Networking Conference 2005 (WCNC 2005), Vol. 4, pp. 2100 - 2105, New Orleans, LA, USA, 13-17 March, 2005.

Bibliography

- [ABCP06] M. F. Abdalla, E. Bresson, O. Chevassut, and D. Pointcheval. Password-based group key exchange in a constant number of rounds. In *Proceeding of PKC 2006*, LNCS 3958, pages 427–442, April 2006.
- [ACP05] M. Abdalla, O. Chevassut, and D. Pointcheval. One-time verifier-based encrypted key exchange. In *Proceedings of the 8th International Workshop on Theory and Practice in Public Key (PKC '05)*, LNCS 3386, pages 47–64, January 2005.
- [ACPR99] J. R. Agre, L. P. Clare, G. J. Pottie, and N. P. Romanov. Development platform for self-organizing wireless sensor networks. In *Proceeding of Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Vol. 3713*, pages 257–268, July 1999.
- [AFP06] M. F. Abdalla, P.-A. Fouque, and D. Pointcheval. Password-based authenticated key exchange in the three-party setting, *IEEE Proceeding-Inf Secur.*, Vol. 153, No. 1, pages 27–39, March 2006.
- [AKNR⁺04] Y. Amir, Y. Kim, C. Nita-Rotaru, John Schultz, Jonathan Stanton, and G. Tsudik. On the performance of group key agreement protocols, *ACM Transactions on Information and Systems Security (TISSEC)*, Vol. 7, No. 3, pages 1–32, August 2004.
- [AN97] T. Aura and P. Nikander. Stateless connections. In *Proceeding of First International Information and Computer Security Conference*, LNCS 1334, pages 87–97, November 1997.
- [AP05] M. Abdalla and D. Pointcheval. Simple password-based encrypted key exchange protocols. In *Topics in Cryptology – CT-RSA 2005*, LNCS 3376, pages 191–208, February 2005.
- [ASSC02] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci. Wireless sensor networks: a survey. *Computer Networks*, Vol. 38, No. 4, pages 393–422, March 2002.
- [AST00] G. Ateniese, M. Steiner, and G. Tsudik. New multiparty authentication services and key agreement protocol. *IEEE Journal on Selected Areas in Communications*, Vol. 18, No. 4, pages 628–639, April 2000.
- [ASWZ02] W. A. Arbaugh, N. Shankar, Y.C.J. Wan, and Kan Zhang. Your 802.11 networks has no clothes. *IEEE Wireless Communications*, Vol. 9, No. 6, pages 44–51, December 2002.
- [AWW05] I.F. Akyildiz, X. Wang, and W. Wang. Wireless mesh networks: A survey. *Computer Networks*, Vol. 47, No. 4, pages 445–487, March 2005.

- [BC04] E. Bresson and D. Catalano. Constant round authenticated group key agreement via distributed computation. In *Proceedings 7th International Workshop on Practice and Theory in Public Key Cryptography PKC'04*, LNCS 2947, pages 115 – 129, March 2004.
- [BCEP04] E. Bresson, O. Chevassut, A. Essiari, and D. Pointcheval. Mutual authentication and group key agreement for low-power mobile devices. *Journal of Computer Communications*, Vol. 27, No. 17, pages 1730–1737, July 2004.
- [BCK96] Mihir Bellare, Ran Canetti, and Hugo Krawczyk. Message authentication using hash functions: the HMAC construction. *CryptoBytes*, Vol. 2, No. 1, pages 12–15, Spring 1996.
- [BCP01] E. Bresson, O. Chevassut, and D. Pointcheval. Provably authenticated group diffie-hellman key exchange - the dynamic case. In *Proceedings of Asiacrypt '01*, LNCS 2248, pages 290–309, December 2001.
- [BCP02a] E. Bresson, O. Chevassut, and D. Pointcheval. Dynamic group diffie-hellman key exchange under standard assumptions. In *Proceedings of Advances in Cryptology - EUROCRYPT '02*, LNCS 2332, pages 321–336, April 2002.
- [BCP02b] E. Bresson, O. Chevassut, and D. Pointcheval. Group diffie-hellman key exchange secure against dictionary attacks. In *Proceeding of Advances in Cryptology - ASIACRYPT 2002*, LNCS 2501, pages 497 – 514, December 2002.
- [BCP07] E. Bresson, O. Chevassut, and D. Pointcheval. Provably secure authenticated group diffie-hellman key exchange. *ACM Trans. Inf. Syst. Secur.*, Vol. 10, No. 3, pages 1094–9224, July 2007.
- [BCPQ01] E. Bresson, O. Chevassut, D. Pointcheval, and J.-J. Quisquater. Provably authenticated group diffie-hellman key exchange. In *Proceedings of ACM CCS '01*, pages 255–264, November 2001.
- [BD95] M. Burmester and Y. Desmedt. A secure and efficient conference key distribution system. In *Advances in Cryptography - Eurocrypt '94*, LNCS 950, pages 275–286, May 1995.
- [BF03] D. Boneh and M. Franklin. Identity-based encryption from the weil pairing. *SIAM Journal of Computing*, Vol. 32, No. 3, pages 585–615, March 2003.
- [BGW01] N. Borisov, I. Goldberg, and D. Wagner. Intercepting mobile communications: the insecurity of 802.11. In *Proceedings of MobiCom '01*, pages 180–189, July 2001.
- [BKR94] M. Bellare, J. Kilian, and P. Rogaway. The security of cipher block chaining. In *Proceedings of CRYPTO '94*, LNCS 839, pages 341–358, August 1994.
- [BL96] D. Boneh and R. J. Lipton. Algorithms for black box fields and their application to cryptography. In *Proceedings of Crypto '96*, LNCS 1109, pages 283–297, August 1996.
- [BLLH06] J. W. Byun, S.-M. Lee, D. H. Lee, and D. Hong. Constant-round password-based group key generation for multi-layer ad-hoc networks. In *Proceeding of SPC 2006*, LNCS 3934, pages 3–17, April 2006.

- [BM03] C. Boyd and A. Mathuria. *Protocols for Authentication and Key Establishment*. 2003.
- [BN03] C. Boyd and J. M. G. Nieto. Round-optimal contributory conference key agreement. In *Proceedings of PKC 2003*, LNCS 2567, pages 161 – 174, January 2003.
- [Boy97] C. Boyd. On key agreement and conference key agreement. In *Proceedings of the Information Security and Privacy: Australasian Conference 1997*, LNCS 1270, pages 294–302, July 1997.
- [BPS⁺04] F. Buiati, R. Puttini, R. De Sousa, C.J. B. Abbas, and L.J. G. Villalba. Authentication and autoconfiguration for manet nodes. In *Proceedings of EUC 2004*, LNCS 3207, pages 41–52, August 2004.
- [BR93] M. Bellare and P. Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *Proceedings of First ACM Conference on Computer and Communications Security 1993*, pages 62–73, November 1993.
- [BW98] K. Becker and U. Wille. Communication complexity of group key distribution. In *Proceedings of the 5th ACM Conference on Computer and Communications Security*, pages 1 – 6, November 1998.
- [CART07] D. R. Choudhary, D. Anshul, S. Roy, and C. S. Thejaswi. Computationally and resource efficient group key agreement for ad hoc sensor networks. In *Proceedings of Communication Systems Software and Middleware, 2007. COMSWARE 2007*, pages 1–10, January 2007.
- [Cho05] S. Choi. Denial-of-service resistant multicast authentication protocol with prediction hashing and one-way key chain. In *Proceedings of the Seventh IEEE International Symposium on Multimedia ISM '05*, pages 701 – 706, December 2005.
- [CJL05] J.-C. Chen, M.-C. Jiang, and Y.-W. Liu. Wireless lan security and ieee 802.11i. *IEEE Wireless Communications*, Vol. 12, No. 1, pages 27–36, February 2005.
- [CKM00] D. W. Carman, P. S. Kruss, and B. J. Matt. Constraints and approaches for distributed sensor network security. Technical Report #00-010, NAI Labs, September 2000.
- [CL06] H.-Y. Chien and R.-Y. Lin. Identity-based key agreement protocol for mobile ad-hoc networks using bilinear pairing. In *Proceeding of IEEE International Conference on Sensor Networks, Ubiquitous, and Trustworthy Computing (SUTC'06)*, pages 520–529, June 2006.
- [CNKW05] S. Cho, J. Nam, S. Kim, and D. Won. An efficient dynamic group key agreement for low-power mobile devices. In *Proceeding of International Conference on Computational Science and Its Applications - ICCSA 2005*, LNCS 3480, pages 498 – 507, May 2005.
- [Cry] Crypto++ Library 5.5.1 Library Benchmark. <http://www.cryptopp.com/benchmarks.html>.
- [CSW03] Y. Chen, E. G. Sirer, and S. B. Wicker. On selection of optimal transmission power for ad hoc networks. In *Proceedings of the 36th Annual Hawaii International Conference on System Sciences (HICSS'03) - Track 9*, pages 1–10, January 2003.

- [dae02] Advanced Encryption Standard, 2002. <http://csrc.nist.gov/encryption/aes/>.
- [DB05] R. Dutta and R. Barua. Constant round dynamic group key agreement. In *Proceedings of Information Security Conference 2005 - ISC 2005*, LNCS 3650, pages 74–88, September 2005.
- [DB06] R. Dutta and R. Barua. Password-based encrypted group key agreement. *International Journal of Network Security*, Vol. 3, No. 1, pages 30–41, July 2006.
- [DBP96] H. Dobbertin, A. Bosselaers, and B. Preneel. RIPEMD-160: A strengthened version of RIPEMD. *Fast Software Encryption*, pages 71–82, April 1996.
- [DH76] W. Diffie and M. Hellman. New directions in cryptography. *IEEE Transaction on Information Theory*, Vol 22, No. 6, pages 644–654, November 1976.
- [DH03] N. Doraswamy and D. Harkins. *IPsec: the new security standard for the Internet, Intranets and Virtual Private Networks (second edition)*. Prentice Hall PTR, 2003.
- [DMS99] L. Dondeti, S. Mukherjee, and A. Samal. A distributed group key management scheme for secure many-to-many communication. Technical Report, PINTL-TR-207-99, Department of Computer Science, University of Maryland, 1999.
- [DPSW06] Y. Desmedt, J. Pieprzyk, R. Steinfeld, and H. X. Wang. A non-malleable group key exchange protocol robust against active insiders. In *Proceedings of ISC 2006*, LNCS 4176, pages 459–475, September 2006.
- [DR00] J. Daemen and V. Rijmen. Rijndael for AES. *AES Candidate Conference*, pages 343–348, April 2000.
- [DSA94] National Institute of Standards and Technology, *Digital Signature Standard*, FIPS Publication 186, 1994.
- [DSL06] D. Damodaran, R. Singh, and Phu Dung Le. Group key management in wireless networks using session keys. In *Proceeding of ITNG 2006*, pages 402–407, April 2006.
- [Dut07] R. Dutta. Multi-party key agreement in password-based setting. In *Proceeding of First Asia International Conference on Modelling & Simulation, 2007. AMS '07.*, pages 133–138, March 2007.
- [ErJ01] D. Eastlake 3rd and P. Jones. US Secure Hash Algorithm 1 (SHA1). RFC 3174, September 2001.
- [FJL00] M. Frodigh, P. Johansson, and P. Larsson. Wireless ad hoc networking - the art of networking without a network. *Ericsson Review*, No. 4, pages 248–263, 2000.
- [FKL⁺05] S. Frankel, K. Kent, R. Lewkowski, A.D. Orebaugh, R.W. Ritchey, and S.R. Sharma. Guide to ipsec vpns. *NIST Special Publication 800-77 (Draft)*, January 2005.
- [FL02] C.-K. Fung and M. C. Lee. A denial-of-service resistant public-key authentication and key establishment protocol. In *Proceeding of IPCCC 2002*, pages 171–178, April 2002.
- [FN01] L. Feeney and M. Nilsson. Investigating the energy consumption of a wireless network interface in an ad hoc networking environment. In *Proceeding of IEEE INFOCOM 2001*, pages 1548–1557, April 2001.

- [FNT⁺06] N. Funabika, T. Nakanishi, H. Takahashi, K. Miki, and J. Kawashima. A proposal for anonymous iee 802.1x authentication protocol for wireless networks. In *Proceedings of Second Workshop on Secure Network Protocols (NPSec) 2006*, November 2006.
- [For02] ATM Forum. *Private Network to Network Interface Specification Version 1.1*, April 2002.
- [Gam85] T. El Gamal. A public key cryptosystem and a signature scheme based on discrete logarithms, *IEEE Trans. Inform. Theory*, Vol. 31, No. 4, pages 469–472, July 1985.
- [GK01] J. Go and K. Kim. Wireless authentication protocol preserving user anonymity. In *Proceeding of SCIS 2001*, pages 159–164, January 2001.
- [GM84] S. Goldwasser and S. Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, Vol. 28, No. 2, pages 270–299, April 1984.
- [GPR⁺06] S.K. Ghosh, R.K. Patro, M. Raina, C. Thejaswi, and V. Ganapathy. Secure group communication in wireless sensor networks. In *Proceeding of 1st International Symposium on Wireless Pervasive Computing 2006*, pages 1–5, January 2006.
- [GQ88] L. C. Guillou and J. J. Quisquater. A “paradoxical” identity-based signature scheme resulting from zero-knowledge. In *Proceedings of Advances in Cryptology - Crypto '88*, LNCS 0403, pages 216–231, August 1988.
- [GS95] L. Gong and P. Syverson. Fail-stop protocols: An approach to designing secure protocols. In *Proceedings of the 5th International Working Conference on Dependable Computing for Critical Applications (DCCA-5)*, pages 44–55, September 1995.
- [GS06] R. C. Gangwar and A. K. Sarje. Secure and efficient dynamic group key agreement protocol for an ad hoc network. In *Proceedings of Ad Hoc and Ubiquitous Computing, 2006. ISAUHC '06*, pages 56–61, December 2006.
- [GST07] J. Groszschädl, A. Szekely, and S. Tillich. The energy cost of cryptographic key establishment in wireless sensor networks. In *Proceedings of the 2nd ACM Symposium on Information, Computer and Communications Security (ASIACCS 2007)*, pages 380 – 382, March 2007.
- [HNY07] L. Hu, X. Niu, and Y. Yang. An efficient multi-server password authenticated key agreement scheme using smart cards. In *Proceedings of International Conference on Multimedia and Ubiquitous Engineering, 2007. MUE '07*, pages 903–907, April 2007.
- [HRM02] T. Halonen, J. Romero, and J. Melero. *GSM, GPRS and EDGE Performance*. Wiley, 2002.
- [HSWC00] W. Heinzelman, A. Sinha, A. Wang, and A. Chandrakasan. Energy-scalable algorithms and protocols for wireless microsensor networks. In *Proceeding of International Conference on Acoustics, Speech, and Signal Processing (ICASSP '00)*, pages 3722–3725, June 2000.
- [HV02] A. Hodjat and I. Verbauwhede. The energy cost of secrets in ad-hoc networks. In *Proceedings of IEEE CAS Workshop on Wireless Communication and Networking*, pages 1–4, September 2002.

- [HWM⁺06] A.M. Hegland, E. Winjum, S.F. Mjølunes, C. Rong, Ø. Kure, and P. Spilling. A survey of key management in ad hoc networks. *IEEE Communications Surveys & Tutorials*, Vol. 8, No. 3, pages 48–66, 3rd Quarter 2006.
- [IEEa] IEEE 802.11i/D4.1, Wireless medium access control (MAC) and physical layer (PHY) specifications:Medium Access Control (MAC) Security Enhancements. July 2003.
- [IEEb] LAN MAN Standards Committee of the IEEE Computer Society, Standards for Local and Metropolitan Area Networks: Standard for Port based network network access control. IEEE Draft P802.1X/D11, March 2001.
- [IEEc] LAN MAN Standards Committee of the IEEE Computer Society, Wireless LAN medium access control (MAC) and physical layer (PHY) specification. IEEE Standard 802.11 1997 Edition, 1997.
- [IEE99] IEEE P1363/D1, Standard Specification for Public-Key Cryptography, November 1999.
- [Int] Intel StrongARM SA-1110 Microprocessor Datasheet.
- [ITW82] I. Ingemarsson, D. T. Tang, and C. K. Wong. A conference key distribution system. *IEEE Transactions on Information Theory*, Vol. 28, No. 5, pages 714–720, September 1982.
- [Jab96] D. P. Jablon. Strong password-only authenticated key exchange. *Computer Communication Review*, Vol. 26, No. 5, pages 5–26, October 1996.
- [JB99] A. Juels and J. G. Brainard. Client puzzles: A cryptographic countermeasure against connection depletion attacks. In *Proceedings of NDSS '99 (Networks and Distributed Security Systems)*, pages 151–165, February 1999.
- [JCKY05] N.-S. Jho, J. H. Cheon, M.-H. Kim, and E. S. Yoo. Broadcast encryption scheme π . Cryptology ePrint Archive: Report 2005/073, March 2005.
- [JM99] D. Johnson and A. Menezes. The elliptic curve digital signature algorithm (ECDSA). Univ. of Waterloo, 1999.
- [JV96] M. Just and S. Vaudenay. Authenticated multi-party key agreement. In *Proceedings of Advances in Cryptology - Asiacrypt '96*, LNCS 1163, pages 36–49, November 1996.
- [KBC97] H. Krawczyk, M. Bellare, and R. Canetti. HMAC: Keyed-Hashing for Message Authentication. RFC 2104, February 1997.
- [KCY⁺02] L. Krishnamurthy, S. Conner, M. Yarvis, J Chhabra, C. Ellison, C. Brabenac, and E. Tsui. Meeting the demands of the digital home with high-speed multi-hp wireless networks. *Intel Technology Journal*, Vol. 6, No. 4, pages 57–68, 2002.
- [KJL06] J. O. Kwon, I. R. Jeong, and D. H. Lee. Provably-secure two-round password-authenticated group key exchange in the standard model. In *Proceeding of IWSEC 2006*, LNCS 4266, pages 322–336, October 2006.

- [KK07] D.-W. Kwak and J. Kim. A decentralized group key management scheme for the decentralized p2p environment. *IEEE Communications Letters*, Vol. 11, No. 6, pages 555–557, June 2007.
- [KLL04] H.-J. Kim, S.-M. Lee, and D.-H. Lee. Constant-round authenticated group key exchange for dynamic groups. In *Proceedings of AsiaCrypt 2004*, LNCS 3329, pages 245 – 259, December 2004.
- [KM03] R. Karri and P. Mishra. Optimizing the energy consumed by secure wireless sessions: wireless transport layer security case study. *Mobile Networks and Applications*, Vol. 8, No. 2, pages 177 – 185, April 2003.
- [KMWZ98] N. Koblitiz, A. J. Menezes, Y.-H. Wu, and R. J. Zuccherato. *Algebraic aspects of cryptography*. 1998.
- [KN93] J. Kohl and C. Neuman. The kerberos network authentication service (v5), 1993.
- [Kob87] N. Koblitiz. Elliptic curve cryptosystems. *Mathematics of Computation*, Vol. 48, No. 177, pages 203–209, 1987.
- [KPT00] Y. Kim, A. Perrig, and G. Tsudik. Simple and fault-tolerant key agreement for dynamic collaborative groups. In *Proceedings of ACM CCS 2000*, pages 235 – 244, November 2000.
- [KPT01] Y. Kim, A. Perrig, and G. Tsudik. Communication-efficient group key agreement. In *Proceedings of IFIP SEC 2001*, pages 229 – 244, June 2001.
- [KPT04a] Y. Kim, A. Perrig, and G. Tsudik. Group key agreement efficient in communication. *IEEE Transactions on Computers*, Vol. 53, No. 7, pages 905–921, July 2004.
- [KPT04b] Y. Kim, A. Perrig, and G. Tsudik. Tree-based group key agreement. *ACM Transaction on Information and System Security*, Vol. 7, No. 1, pages 60 – 96, February 2004.
- [KS99] P. Karn and W. Simpson. Photuris: Session-key management protocol. RFC 2522, 1999.
- [KY03] J. Katz and M. Yung. Scalable protocols for authenticated group key exchange. In *Proceedings of Advances in Cryptography - Crypto'03*, LNCS 2729, pages 110 – 125, August 2003.
- [LF03] M. C. Lee and C.-K. Fung. A public-key based authentication and key establishment protocol coupled with a client puzzle. *Journal of the American Society for Information Science and Technology*, Vol. 54, No. 9, pages 810–823, 2003.
- [LH01] L. Li and J. Halpern. Minimum energy mobile wireless networks revisited. In *Proceeding IEEE International Conference on Communications (ICC) 2001*, pages 278–283, June 2001.
- [LHL04] S.-M. Lee, J. Y. Hwang, and D. H. Lee. Efficient password-based group key exchange. In *Proceeding of TrustBus 2004*, LNCS 3184, pages 191–199, September 2004.

- [LLY06] P.P.C. Lee, J.C.S. Lui, and D.K.Y. Yau. Distributed collaborative key agreement and authentication protocols for dynamic peer groups. *IEEE/ACM Transactions on Networking*, Vol. 14, No. 2, pages 263 – 276. April 2006.
- [LM90] X. Lai and J. Massey. A proposal for a new block encryption standard. In *Proceedings of EUROCRYPT '90*, LNCS 473, pages 389–404, May 1990.
- [LM06] L.J. Liao and M. Manulis. Tree-based group key agreement framework for mobile ad-hoc networks. In *Proceeding of AINA 2006, Vol. 2*, pages 5 – 9, April 2006.
- [LMQ⁺03] L. Law, A. Menezes, M. Qu, J. Solinas, and S. Vanstone. An efficient protocol for authenticated key agreement. *Des. Codes Cryptography*, Vol. 28, No. 2, pages 119–134, 2003.
- [LP07] L. Lazos and R. Poovendran. Power proximity based key management for secure multicast in ad hoc networks, *Wireless Networks*, Vol. 13, No. 1, pages 127–148, 2007.
- [LRDP02] K. Lahiri, A. Raghunathan, S. Dey, and D. Panigrahi. Battery-driven system design: A new frontier in low power design. In *Proceedings of Asia South Pacific Design Automation Conference (ASP-DAC)/International Conference on VLSI Design*, pages 261–267, January 2002.
- [LTL⁺05] J. Liu, B. Tang, Y. Liu, C. Liu, and Y.-'A. Liu. A password based authentication protocol for access control in WLAN. In *Proceeding of ISCIT 2005*, pages 1136–1139, October 2005.
- [Luc] Lucent Technologies. "WaveLan IEEE 802.11 PC Card User's Guide". February, 1999.
- [Ma05] M. Ma. Mitigating denial of service attacks with password puzzles. In *Proceedings of ITCC'05*, pages 621– 626, April 2005.
- [Mac02] P. D. MacKenzie. The pak suite: Protocols for password-authenticated key exchange. Technical Report. 2002-46, DIMACS Center, Rutgers University, 2002.
- [Mea99] C. Meadows. A formal framework and evaluation method for network denial of service. In *Proceedings of the 12th IEEE workshop on Computer Security Foundations (CSFW '99)*, pages 4–13, June 1999.
- [Men93] A. J. Menezes, editor. *Elliptic Curve Public Key Cryptosystems*. Kluwer Academic Publishers, 1993.
- [Mil86] V. S. Miller. Use of elliptic curves in cryptography. In *Proceedings of CRYPTO '85*, LNCS 218, pages 417–426, August 1986.
- [Mir] MIRACL: Multiprecision Integer and Rational Arithmetic C/C++ Library. <http://indigo.ie/mscott/>.
- [Moo98] G. E. Moore. Cramming more components onto integrated circuits. *Proceedings of the IEEE*, Vol. 86, No. 1, pages 82–85. January 1998.
- [Mot05] Motorola. Mesh Networks, Motorola Techonology Position paper (White paper), 2005.

- [MOV97] A. J. Menezes, P. C. V. Oorschot, and S. A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 1997.
- [MPS00] P. MacKenzie, S. Patel, and R. Swaminathan. Password-authenticated key exchange based on RSA. In *Proceedings of Asiacrypt 2000*, LNCS 1976, pages 599–613, December 2000.
- [Nat93] National Institute of Standards and Technology (NIST). *FIPS Publication 46-2: Data Encryption Standard*, December 30, 1993.
- [NHSC07] W. H. D. Ng, M. Howarth, Z. Sun, and H. Cruickshank. Dynamic balanced key tree management for secure multicast communications. *IEEE Transactions on Computers*, Vol. 56, No. 5, pages 590–605, May 2007.
- [Niu07] Q. Niu. Study and implementation of a improved group key protocol for mobile ad hoc networks. In *Proceedings of Eighth ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing, 2007 (SNPD 2007)*, pages 304–308, August 2007.
- [NKW05] J. Nam, S. Kim, and D. Won. Secure group communications over combined wired and wireless networks. In *Proceeding of TrustBus 2005*, LNCS 3592, pages 90–99, August 2005.
- [NLK⁺04] J. Nam, J. Lee, S. Kim, S. Kim, and D. Won. Provably-secure and communication-efficient protocol for dynamic group key exchange. *Journal of the KIISC (Korean Institute of Information Security and Cryptology)*, Vol. 14, No. 4, pages 163–181, August 2004.
- [NLKW04] J. Nam, J. Lee, S. Kim, and D. Won. DDH-based group key agreement for mobile computing. Cryptology ePrint Archive, Report 2004/127, May 2004.
- [NLKW05] J. Nam, J. Lee, S. Kim, and D. Won. DDH-based group key agreement in a mobile environment, *J. Syst. Softw.*, Vol. 78, No. 1, pages 73–83, October 2005.
- [oS95] National Bureau of Standards. Secure hash standard. Technical Report FIPS Publication 180-1, National Bureau of Standards, 1995.
- [PL00] J. Pieprzyk and C. H. Li. Multiparty key agreement protocols. *IEE Proceedings - Computers and Digital Techniques*, Vol. 147, No. 4, pages 229–236, July 2000.
- [PLP06] K. Piotrowski, P. Langendoerfer, and S. Peter. How public key cryptography influences wireless sensor node lifetime. In *Proceedings of ACM SASN 2006*, pages 169–176, October 2006.
- [PMB06] B. Panja, S.K. Madria, and B. Bhargava. Energy and communication efficient group key management protocol for hierarchical sensor networks. In *Proceeding of IEEE International Conference on Sensor Networks, Ubiquitous, and Trustworthy Computing 2006*, pages 384–393, June 2006.
- [PMW⁺01] C. Perkins, J. T. Malinen, R. Wakikawa, E. M. Belding-Royer, and Y. Sun. IP address autoconfiguration for ad hoc networks. IETF draft, 2001.

- [PRRJ03] N. Potlapally, S. Ravi, A. Raghunathan, and N. Jha. Analyzing the energy consumption of security protocols. In *Proceedings of the 2003 International Symposium of Low Power Electronics and Design (ISLPED '03)*, pages 30–35, August 2003.
- [PRRJ06] N. R. Potlapally, S. Ravi, A. Raghunathan, and N. K. Jha. A study of the energy consumption characteristics of cryptographic algorithms and security protocols. *IEEE Transactions in Mobile Computing*, Vol. 5, No. 2, pages 128–143, February 2006.
- [Rab79] M. Rabin. Digital signatures and public key functions as intractable as factorization. *MIT Laboratory for Computer Science, Technical report MIT/LCS/TR-212*, January 1979.
- [RBD99] O. Rodeh, K. Birman, and D. Dolev. Optimized group rekey for group communication systems. *Technical Report: TR99-1764, Cornell University*, 1999.
- [Riv92] Ronald L. Rivest. The MD5 message-digest algorithm. RFC 1321, April 1992.
- [Riv95] Ronald L. Rivest. The RC5 encryption algorithm. In Bart Preneel, editor, *Fast Software Encryption*, pages 86–96, 1995. (Proceeding Second International Workshop, Dec. 1994, Leuven, Belgium).
- [RSA78] R. L. Rivest, A. Shamir, and L. M. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, Vol. 21, No. 2, pages 120–126, February 1978.
- [RWRS00] C. Rigney, S. Willens, A. Rubens, and W. Simpson. Remote authentication dial in user service (RADIUS), RFC 2865, June 2000.
- [SBL⁺04] M. Singhal, R. Bai, Y. Lin, Y. Wang, M. Yang, and Q Zhang. Key management protocols for wireless networks. Lab for Advanced Networking, Dept of Computer Science, University of Kentucky. Technical Report, 2004.
- [SC01] A. Sinha and A. Chandrakasan. Jouletrack - a web based tool for software energy profiling. In *Proceedings of Design Automation Conference 2001*, pages 220–225, June 2001.
- [Sch91] C. P. Schnorr. Efficient signature generation by smart cards. *Journal of Cryptology*, Vol. 4, No. 3, pages 161–174. 1991.
- [Sch94] Bruce Schneier. The blowfish encryption algorithm. *Dr. Dobbs's Journal*, pages 38–40, April 1994.
- [SCI⁺01] E. Shih, S.-H. Cho, N. Ickes, R. Min, A. Sinha, A. Wang, and A. Chandrakasan. Physical layer driven protocol and algorithm design for energy-efficient wireless sensor networks. In *Proceedings of ACM SIGMOBILE 2001*, pages 272 – 287, July 2001.
- [SFW⁺01] S. Schmid, J. Finney, M. Wu, A. Friday, A. C. Scott, and W. D. Shepherd. An access control architecture for microcellular wireless ipv6 networks. In *Proceeding of Local Computer Networks (LCN) 2001*, pages 454–463, November 2001.
- [SH06] N. B. Salem and J.-P. Hubaux. Securing wireless mesh networks. *IEEE Wireless Communications*, Vol. 13, No. 2, pages 50–55, April 2006.

- [Sha79] A. Shamir. How to share a secret, *Communications of the ACM*, Vol. 22, No. 11, pages 612–613, November 1979.
- [SKI06] S. H. Shin, K. Kobara, and H. Imai. Elliptic curve based authenticated key agreement protocol for wireless security. In *Proceedings of Computational Intelligence and Security 2006*, pages 1096 – 1100, November 2006.
- [SLS07] J.-H. Son, J.-S. Lee, and S.-W. Seo. Energy efficient group key management scheme for wireless sensor networks. In *Proceedings of 2nd International Conference on Communication Systems Software and Middleware 2007 (COMSWARE 2007)*, pages 1 – 9, January 2007.
- [SNS88] J. G. Steiner, B. C. Neuman, and J. I. Schiller. Kerberos: an authentication service for open network systems. In *Proceedings of Usenix Conference*, pages 191–202, February 1988.
- [SOK00] R. Sakai, K. Ohgishi, and M. Kasahara. Cryptosystems based on pairing. In *Proceedings of SCIS 2000*, pages 26–28, January 2000.
- [spe] Spectrum24 High Rate LA 41X1 PC Card spectrum datasheet.
- [SSKV06] T. Srinivasan, S. Sathish, R.V. Kumar, and M.V.B. Vijayender. A hybrid scalable group key management approach for large dynamic multicast networks. In *Proceedings of CIT 2006*, pages 102 – 102, September 2006.
- [SSN98] S. Saeednia and R. Safavi-Naini. Efficient identity-based conference key distribution protocols. In *Proceedings of 3rd Australasian Conference Information Security and Privacy*, LNCS 1438, pages 320–331, July 1998.
- [Sti05] D. Stinson. *Cryptography: Theory and Practice, Third Edition (Discrete Mathematics and Its Applications)*. Chapman & Hall/CRC, 2005.
- [Str06] M.A. Strangio. An optimal round two-party password-authenticated key agreement protocol. In *Proceedings of the First International Conference on ARES 2006*, pages 216 – 223, April 2006.
- [STW96] M. Steiner, G. Tsudik, and M. Waidner. Diffie-hellman key distribution extended to group communication. In *Proceedings of 3rd ACM Conference on Computer and Communications Security*, pages 31 – 37, March 1996.
- [STW98] M. Steiner, G. Tsudik, and M. Waidner. Cliques: A new approach to group key agreement. In *Proceedings of the 18th International Conference on Distributed Computing Systems*, pages 380–387, May 1998.
- [STW00] M. Steiner, G. Tsudik, and M. Waidner. Key agreement in dynamic peer groups. *IEEE Transactions on Parallel and Distributed Systems*, Vol. 11, No. 8, pages 769–780, August 2000.
- [STY04] N. Saxena, G. Tsudik, and J. H. Yi. Identity-based access control for ad hoc groups. In *Proceedings of ICISC 2004*, LNCS 3506, pages 362–379, December 2004.

- [Tse05] Y.-M. Tseng. Efficient authenticated key agreement protocols resistant to a denial-of-service attack. *International Journal of Network Management*, Vol. 15, No. 3, pages 193–202, May 2005.
- [Tse07] Y.-M. Tseng. A secure authenticated group key agreement protocol for resource-limited mobile devices. *The Computer Journal*, Vol. 50, No. 1, pages 41–52, 2007.
- [TT00] W.-G. Tzeng and Z.-J. Tzeng. Round-efficient conference-key agreement protocols with provable security. In *Proceedings of Advances in Cryptology - ASIACRYPT 2000*, LNCS 1976, pages 614 – 618, December 2000.
- [TWL05] W. Trappe, Y. Wang, and K. J. R. Liu. Resource-aware conference key establishment for heterogeneous networks. *IEEE/ACM Transactions on Networking*, Vol. 13, No. 1, pages 134–146, February 2005.
- [VM97] V. Varadharajan and Y. Mu. Preserving privacy in mobile communications: A hybrid method. In *Proceeding of IEEE International Conference on Personal Wireless Communications 1997*, pages 532–536, December 1997.
- [Wan04] S.-J. Wang. Anonymous wireless authentication on a portable cellular mobile system. *IEEE Transaction on Computers*, Vol. 53, No. 10, pages 1317–1329, October 2004.
- [WDBA05] Z. G. Wan, R. H. Deng, F. Bao, and A. L. Ananda. Anonymous dos-resistant access control protocol using passwords for wireless networks. In *Proceeding of Local Computer Networks (LCN) 2005*, pages 328–335, November 2005.
- [Wen05] K. Weniger. PACMAN: Passive autoconfiguration for mobile ad hoc networks. *IEEE Journal On Selected Areas in Communications*, Vol. 23, No. 3, pages 507–519. March 2005.
- [WFH07] N.-C. Wang, S.-Z. Fang, and Y.-F. Huang. Hierarchy-based key management for secure group communications in mobile ad hoc networks. In *Proceedings of the 2007 international conference on Wireless communications and mobile computing (IWCMC '07)*, pages 571–576, August 2007.
- [WGL00] C. K. Wong, M. G. Gouda, and S. S. Lam. Secure group communications using key graphs. *IEEE/ACM Transactions on Networking*, Vol. 8, No. 1, pages 16–30, 2000.
- [WHA98] D. M. Wallner, E. G. Harder, and R. C. Agee. Key management for multicast: Issues and architecture. *internet draft draft-wallner-key-arch-01.txt*, September 1998.
- [WJHF04] B. Waters, A. Juels, J. A. Halderman, and E. W. Felten. New client puzzle outsourcing techniques for dos resistance. In *Proceedings of the 11th ACM conference on Computer and communications security (CCS '04)*, pages 246–256, October 2004.
- [Won05] D. S. Wong. Security analysis of two anonymous authentication protocols for distributed wireless networks. In *Proceeding of PerCom 2005 Workshops*, pages 284 – 288, March 2005.
- [Wu98] T. Wu. The secure remote password protocol. In *Proceedings of the 1998 Internet Society Network and Distributed System Security Symposium*, pages 97–111, March 1998.

- [WZ04] K. Weniger and M. Zitterbart. Address autoconfiguration in mobile ad hoc networks: Current approaches and future directions. *IEEE Network Magazine*, Vol. 18, No. 4, pages 6–11, 2004.
- [WZ06] S. H. Wu and Y. F. Zhu. Efficient and secure password-based authenticated key exchange protocol. In *Proceeding of Computational Intelligence and Security 2006*, pages 1269 – 1272, November 2006.
- [YFH07] G. Yao, D. Feng, and X. Han. Improved client-to-client password-authenticated key exchange protocol. In *Proceedings of The Second International Conference on Availability, Reliability and Security, 2007. ARES 2007*, pages 564–574, April 2007.
- [YSL07] W. Yu, L. Y. Sun, and R. K. J. Liu. Optimizing rekeying cost for contributory group key agreement schemes. *IEEE Transactions on Dependable and Secure Computing*, Vol. 4, No. 3, pages 228–242, July-Sept. 2007.
- [YY06] E.-J. Yoon and K.-Y. Yoo. Secure password authentication protocol in wireless networks. In *Proceeding of NWeSP 2006*, pages 149 – 154, September 2006.
- [ZR04] L. Zhou and C. V. Ravishankar. Efficient, authenticated, and fault-tolerant key agreement for dynamic peer groups. In *Proceedings of Third International IFIP-TC6 Networking Conference, NETWORKING 2004*, LNCS 3042, pages 759 – 770, May 2004.