ON SECURE GROUP KEY AGREEMENT PROTOCOLS FOR MOBILE AD HOC NETWORKS

A THESIS

Submitted in partial fulfilment of the requirements for the award of the degree of

DOCTOR OF PHILOSOPHY

in

ELECTRONICS AND COMPUTER ENGINEERING

by

RAKESH CHANDRA GANGWAR

DEPARTMENT OF ELECTRONICS AND COMPUTER ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY ROORKEE
ROORKEE - 247 667 (INDIA)

FEBRUARY, 2009

INDIAN INSTITUTE OF TECHNOLOGY ROORKEE
ROORKEE

CANDIDATE'S DECLARATION

I hereby certify that the work which is being presented in the thesis entitled **"ON SECURE GROUP KEY AGREEMENT PROTOCOLS FOR MOBILE AD HOC NETWORKS"** in partial fulfilment of the requirements for the award of the Degree of Doctor of Philosophy and submitted in the Department of Electronics and Computer Engineering of Indian Institute of Technology Roorkee, Roorkee is an authentic record of my own work carried out during a period from July 2004 to February 2009 under the supervision of **Dr. Anil K. Sarje**, Professor, Department of Electronics and Computer Engineering, Indian Institute of Technology Roorkee, Roorkee.

The matter presented in this thesis has not been submitted by me for the award of any other degree of this or any other Institute.

(RAKESH CHANDRA GANGWAR)

This is to certify that the above statement made by the candidate is correct to the best of my knowledge.

Date: 27/04/2009

(Anil K. Sarje)
Supervisor

The Ph.D. Viva-Voce Examination of **Rakesh Chandra Gangwar**, Research Scholar, has been held on ....Nov. 3, 2009

Signature of Supervisor 3/11/09

3.11.2009
Signature of External Examiner

Abstract

Nowadays, the popularity of wireless network is increasing very fast. Wireless technology is gaining more and more attention from both academia and industry. Most wireless networks used today, for example, the cell phone networks and the 802.11 Wireless Local Area Networks (WLANs), are based on the wireless network model with pre-existing wired network infrastructures. Packets from source wireless nodes are received by nearby base stations, then injected into the underlying network infrastructure and finally transferred to destination nodes.

Another wireless network model, which is actively researched, is the mobile ad hoc network (MANET). This network is formed by only mobile nodes and requires no pre-existing network infrastructure. Nodes with wireless capability form a MANET instantly and packets can be delivered to any node in the network. Since there is no base station and no underlying network infrastructure in MANETs, some mobile nodes work as routers to relay packets from source to destination. It is very easy and economic to form an MANET in real time. MANET is ideal in situations like battlefield or emergency rescue area where fixed network infrastructure is very hard to deploy.

In recent years the popularity of multicast and group-oriented applications is also rapidly increasing in both scenarios such as wireless as well as wireline networks. Group communication provides point-to-multi-point or multi-point-to-multi-point communication by organizing processes in groups. A set of such processes is called group, where every process is a member of a group. Multicast and group communications are peer to peer and occur in many different settings, such as software updates, multimedia content distribution, interactive gaming and stock data distribution, voice and video conferencing, shared white boards, distributed simulations, as well as online games, replicated servers and databases of all types. The multicast and group communication systems are, therefore, used as vehicle to provide efficient data delivery to a large audience. One of major concerns for such systems is security. Security services include data confidentiality, integrity, authenticity, access control, etc. Many of these services can be facilitated if all group members share a common group

key. This makes secure and efficient group key creation, distribution and management a fundamental service and design challenge for Group Communication.

The existing group key establishment protocols can be categorized into three categories: centralized, decentralized and contributory. In centralized protocols only one central server is responsible for creation and distribution of keys. In decentralized protocols based on common group key approach, one central server creates and other additional servers help in distributing keys, this way other servers share the load of the central server. In another category of decentralized protocols, the group is divided into subgroups and each subgroup has its own subgroup key. Although this eliminates one-affect-all phenomena, yet it involves a lot of encryption/decryption cost. Since nodes in MANET are not computationally powerful, no one node can work as central server for creating and distributing the keys. Therefore, these protocols are not suitable for secure and efficient group communication in mobile ad hoc networks. In contributory protocols each node contributes equally in group key formation, therefore, these protocols seem to be promising for mobile ad hoc networks. The contributory protocols are also known as group key agreement protocols. This thesis concentrates on group key agreement protocols that are based on the two party Diffie-Hellman protocol.

In the thesis we first analyze various existing group key agreement protocols based on the two party key agreement protocol of W. Diffie and M. Hellman, for example, INGM, BD, Hypercube, Octopus, Clique (IKA.1 and IKA.2), TGDH, NAGKA, and STR. The parameters such as number of rounds, number of messages communicated, number of exponential operations performed and round synchronization are taken into consideration for complexity analysis. All protocols other than Clique protocols need round synchronization, which involves a synchronising device like clock that creates parallel broadcasts, which are problematic in mobile ad hoc networks. From this analysis it is concluded that the Clique protocol suite (IKA.1 and IKA.2) show the best performance among all the protocols considered. Among clique protocols, IKA.2 protocol needs less number of exponential operations as compared to IKA.1 protocol. Although IKA.1 protocol is inferior to IKA.2 in terms of exponential operations, yet it is more efficient regarding the number of messages.

Group communication involves various dynamic events such as join, merge, leave, and partition, therefore, a group key agreement protocol must make efficient provision for forward and backward secrecy, which means that a joining member(s) should not be able to

decrypt the past communications and leaving member(s) should not be able to decrypt the future communications. So we next analyze group key agreement protocols based on two party Diffie-Hellman protocol for above dynamic membership events. We observe that Clique protocols do not need round synchronization among various group key agreement protocols. And IKA.2 protocol shows the better performance for merge, leave and partition events as far as exponential operations are concerned. It is again concluded that Clique protocols suite shows the best results, especially IKA.2, among various group key agreement protocols.

Thus it is observed that a group key agreement protocol must be secure and efficient in various complexity measures. The total number of exponential operations should be small because nodes in MANETs are computationally weak. Further if the number of rounds is independent of group size, the protocol will be highly scalable.

Next we propose a protocol suite, which is natural extension of two party Diffie-Hellman protocol. The proposed protocol not only provides efficient algorithms for setup, join (merge) and leave (partition), but also member authentication service. The proposed protocol also has provisions for all valid members to detect errors in communicated messages and stop execution of the protocol immediately as they encounter invalid message from the members who have awry intentions. This helps in eliminating the man-in-the-middle attack in addition to message corruption due to system faults using message verification.

In the proposed protocol suite the members are arranged in a logical ring. The setup algorithm takes initial group as an input and outputs a group key after second round. At the end of each round members are authenticated, and after second round messages are also verified for corruption. If message validity checks fail the protocol terminates immediately, otherwise a group key can be generated and saved by each member in addition to three other parameters, which are used during the dynamic events, i.e., join, leave, multiple join and multiple leave. After computing the group key and three parameters other ephemeral data is erased by each member.

The proposed protocol suite also consists of an algorithm, which can be used for dynamic events join and merge. The algorithm takes the initial group and joining member(s) as an input and outputs the group key and three other parameters for each member. The algorithm also takes two rounds. After each rounds members are authenticated, and after second round messages are also verified for corruption. If message validity checks fail the

protocol terminate immediately, otherwise a group key can be generated and saved by each member in addition to other three parameters, which are used during the future dynamic events. After computing the group key and three parameters other ephemeral data is erased by each member.

The proposed protocol suite also consists of an algorithm for leave and partition events. The leave-partition algorithm also takes two rounds. After each round members are authenticated, and after second round messages are also verified for its corruption. If message validity checks fail the protocol terminate immediately, otherwise a group key can be generated. After computing the group key and other parameters, ephemeral data is erased by each member.

Security analysis of the proposed protocol has been done in random oracle model. For which a number of oracle queries are used, which can be replaced by actual function for practical purpose, for example, secure signature scheme ($\sum$) and hash function (H) may be replaced by ElGamal digital signature algorithm and secure hash algorithm (SHA-1) respectively. The selected pseudorandom number in ElGamal digital signature algorithm plays an important role in eliminating the impersonation attack when a secret key is somehow compromised.

Further, the work shows the comparison between the proposed protocol and Clique protocol suite. The proposed protocol suite does not involve round synchronization as in case of Clique protocols. Thereby, no synchronous mechanism is needed. The number of rounds in setup protocol of IKA.1 and IKA.2 measures as n and $n+1$ (where n is the number of group members) whereas our protocol needs only two rounds irrespective of the group size. *join-merge* (for join or merge) and *leave-partition* (for leave or partition) protocols also needs two rounds each for their completion, which indicates that the proposed protocol suite is scalable too. The setup protocol in proposed protocol suite needs $3n$ exponential operations whereas IKA.1 and IKA.2 need $(\frac{n}{2}(n+3)-1)$ and $(5n-6)$ exponential operations respectively. However, the number of messages n in IKA.1 protocol is less as compared to IKA.2 and the proposed protocol, which need $2n-1$ and $2n$ messages respectively. It is also seen that the *join-merge* of our protocol is more efficient in number of exponential operations measure as compared to join (merge) protocol of IKA.1 and IKA.2 protocols for large group size. And, the proposed *leave-partition* protocol is also efficient as compared to IKA.1 and IKA.2 when

the n (the number of current group members) is high and l (the leaving members) is low. Our *leave-partition* protocol takes $6l$ exponential operations as compared to $2(n-l)$ and $(n-l)$ exponential operations of IKA.1 and IKA.2 protocol, where l is number of leaving members. However, in case of number of messages, our *leave-partition* protocol is inferior to IKA.1 and IKA.2 protocols. In the proposed setup protocol, the total cost of computations has been reduced considerably. For authentication, each group member generates two signatures and performs $2n$ signature verifications, which creates additional cost for authentication. Other existing protocols, for example, INGM, BD, Hypercube, Octopus, Clique (IKA.1 and IKA.2), TGDH, NAGKA, and STR, do not provide authentication, our protocol suite provides member authentication at a nominal cost. In proposed setup protocol, each group member performs at most 3 exponential operations, 4 one-way hash function operations, and n XOR operations. Since the operation dependent on the number of group members is the XOR operation, this way the total cost of computation has been reduced considerably.

In view of the above security and complexity analysis of proposed protocol suite and various existing group key agreement protocols, it is logically concluded that the proposed protocol suite outperforms INGM, BD, Hypercube, Octopus, TGDH, NAGKA, STR as well as Clique protocols (IKA.1 and IKA.2) in terms of number of rounds, number of messages, number of exponential operations and round synchronization.

Acknowledgements

I would like to express my deepest gratitude to my learned supervisor **Dr. Anil K. Sarje** for his encouragement, painstaking supervision, innovative suggestions and invaluable help during the entire period of my Ph.D. studies. It has been a great honor and pleasure for me to work under his supervision. I learned a great deal from him, not only about research but also about matters touching many other aspects, which will benefit me in future life and career.

The cooperation and help extended by the Head and faculty members, Department of Electronics and Computer Engineering, Indian Institute of Technology Roorkee is gratefully acknowledged. I also wish to thank my research committee members for providing insightful and constructive comments.

I am highly obliged to the Dr. Dial Chand, Principal, Beant College of Engineering and Technology, Gurdaspur (Pb.), for sponsoring me.

I wish to convey my appreciation to my fellow research scholars who have provided me encouragement and timely support.

Finally, I could not reach the important milestone of my life without the support and encouragement of my family. I express my sincere appreciation and gratitude to my wife Dr. Smita Singh and daughters, namely, Srishti & Solaris for their patience and encouragement when it was most required.

And above all, I am thankful to the Almighty whose divine grace gave me the required courage, strength and perseverance to overcome various obstacles that stood in my way.

Rakesh Chandra Gangwar

Contents

List of Abbreviations

PPV	Pay-Per-View
WLAN	Wireless Local Area Network
MANET	Mobile Ad Hoc Network
PAN	Personal Area Network
PDA	Personal Digital Assistant
KEK	Key Encryption Key
GKMP	Group Key Management Protocol
SM	Security Manager
KDC	Key Distribution Center
GTEK	Group Traffic Encryption Key
GKEK	Group Key Encryption Key
SKP	Session Key Packet
STEK	Session Traffic Encryption Key
SKEK	Session Key Encryption Key
GRP	Group Rekey Packet
GKM	Group Key Management
ACL	Access Control List
CA	Certificate Authority
LKH	Logical Key Hierarchy
OFT	One-way Function Trees
ELK	Efficient Large-group Key

PRF	Pseudo-Random Function
CTKM	Centralized Tree-based Key Management
CFKM	Centralized Flat-table Key Management
SMKD	Scalable Multicast Key Distribution
CBT	Core Based Tree
IGKMP	Intra-domain Group Key Management Protocol
DKD	Domain Key Distributor
AKD	Area Key Distributor
HS	Hydra Server
GC	Group Controller
LC	Local Controller
DEP	Dual Encryption Protocol
SGM	Sub Group Manager
SIDS	Session Identities
PIDS	Partner Identities
GDH	Group Diffie-Hellman
IKA.1	Initial Key Agreement 1
IKA.2	Initial Key Agreement 2
TGDH	Tree-based Group Diffie-Hellman
NAGKA	Non-Authenticated Group Key Agreement
AGKA	Authenticated Group Key Agreement
PKI	Public Key Infrastructure
CA	Certificate Authority
AGKA-G	Authenticated Group Key Agreement using Gunther's scheme

STR	Skinny Tree
GK	Group Key
GSA	Group Security Agent
GSC	Group Security Controller
KHIP	Keyed HIerarchical-multicast Protocol
RK	Random Key
CS	Cipher Sequence
CG	Cipher Group
SAKM	Scalable and Adaptive Key Management
KS	Key Server
XOR	Exclusive OR
sk	Secret Key
pk	Public Key
CDH	Computational Diffie-Hellman
MAC	Medium Access Control

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

In the recent years the popularity of multicast and group-oriented applications is rapidly increasing in both scenarios such as wireless and wire-line networks. Multicast and group communications occur in many different settings, such as software updates, multimedia content distribution, interactive gaming and stock data distribution, audio and video conferencing, remote consultation and diagnostics systems for medical applications, contract negotiation, shared white boards, distributed simulations, multi-party games, electronic commerce environments like online real time auctions, replicated servers and databases of all types. Multicast and group communication systems are used as vehicle to provide efficient and secure data delivery to a large audience in such scenario..

Mobile ad hoc networks (MANETs)[10,17,42,44,63,82] have attracted significant attention recently due to their wide applications in different areas. These networks do not have fixed infrastructure, and are useful in applications such as military operations, relief and rescue operation in case of natural disaster etc. MANETs are also very attractive option for commercial uses. The aforementioned multicast and group-oriented network applications can also be conducted in this network environment. The nature of ad-hoc networks sets certain additional requirements for the group key establishment protocols. One major concerns for group communication in mobile ad hoc network is security [2,42,47,54,63,64,97,102]. Security services include data confidentiality, integrity, authentication, access control, etc. Another major concern is efficiency, which is measured in terms of the number of communication rounds, number of messages, number of exponential operations required to create a group key. Many of the above concerns can be addressed if all group members share a common group key in secure and efficient manner. This makes secure and efficient group

key creation, distribution and management a fundamental issue, and a design challenge for secure and reliable group communication in mobile ad hoc network.

1.2 Group communication

Group communication implies point-to-multi-point or multi-point-to-multi-point communication by organizing processes in groups, where every process is a member of a group. To identify and distinguish multiple groups, every group is associated with a unique group name or IP–multicast address [94]. For instance, a group may consist of users playing an online game with each other. Another group may consist of members participating in a multimedia conference. In these examples, each group member sends a message targeted to a particular group. The group communication service then delivers the message to all other group members. Groups are usually dynamic in the sense that the list of group members continuously changes. Users can choose when they wish to join or leave a group, for example, users can independently start or stop playing a game at any time. Therefore, groups can be seen as dynamic sets of entities.

1.3 Types of Group communication

As mentioned before, groups consist of multiple members exchanging messages between themselves. Depending on how messages are exchanged, the literature distinguishes two different types of group communication as briefly described below.

1.3.1 One-to-many group communication

This type of a group communication, also referred to as **"1–to–N multicast"**, is a unidirectional transmission from one sender to multiple recipients. This means that only one entity sends information and multiple entities receive it [1,8,52,60,74], as shown in Fig 1.1. Since in this type of group communication only one entity can send data, there is no information that can be received by the sender [79]. Therefore, the sender is usually the initiator of the group and is not considered a group member. Examples of this type of multicasting include all Pay-Per-View (PPV) applications, transmission of passive data (e.g. stock market information, web–radio channels, software update distribution etc.) and others. Typically, PPV multicast contains monetary information where receivers have to pay for it. Thus, this type of group communication in combination with security services is of interest to

commercial companies in order to guarantee payment by consumers. This, for instance, could require the consumer to give his credit card in order to obtain the key and to get access to monetary information.

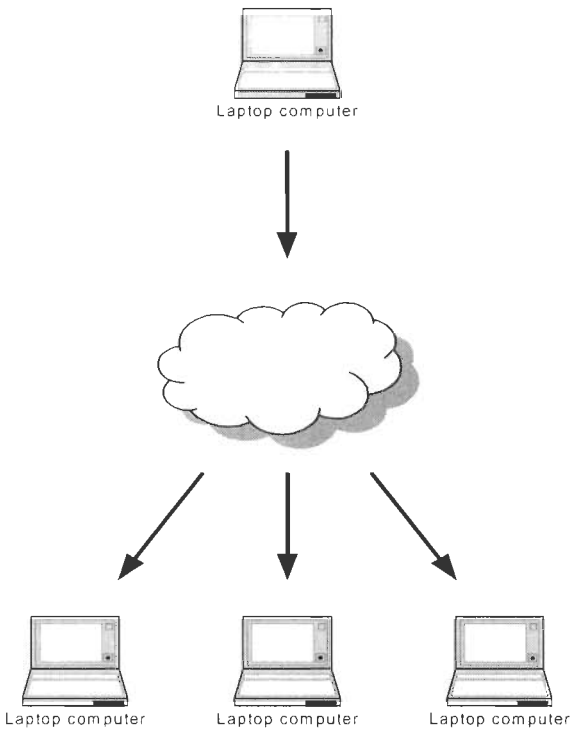


Figure 1.1: One-to-many group communication

1.3.2 Many-to-many group communication

Another type of group communication among multiple entities is the situation where every member can be both, a sender and receiver, as shown in Fig 1.2. This type is also referred to as **"N–to–N communication"** [3-5,9,12,13,15,16,18,19,21,22,25,27-33,35,36,38-41,46,50,53,57-59,61,62,65,66,68,71,74-78,84-93,96,98,99,104,105,107,110,112-116]. The main difference compared with the one-to-many is that all entities involved in a group communication are equivalent. Typical examples for this type of group communication are a video conferencing among multiple members or an application for exchanging messages and documents.

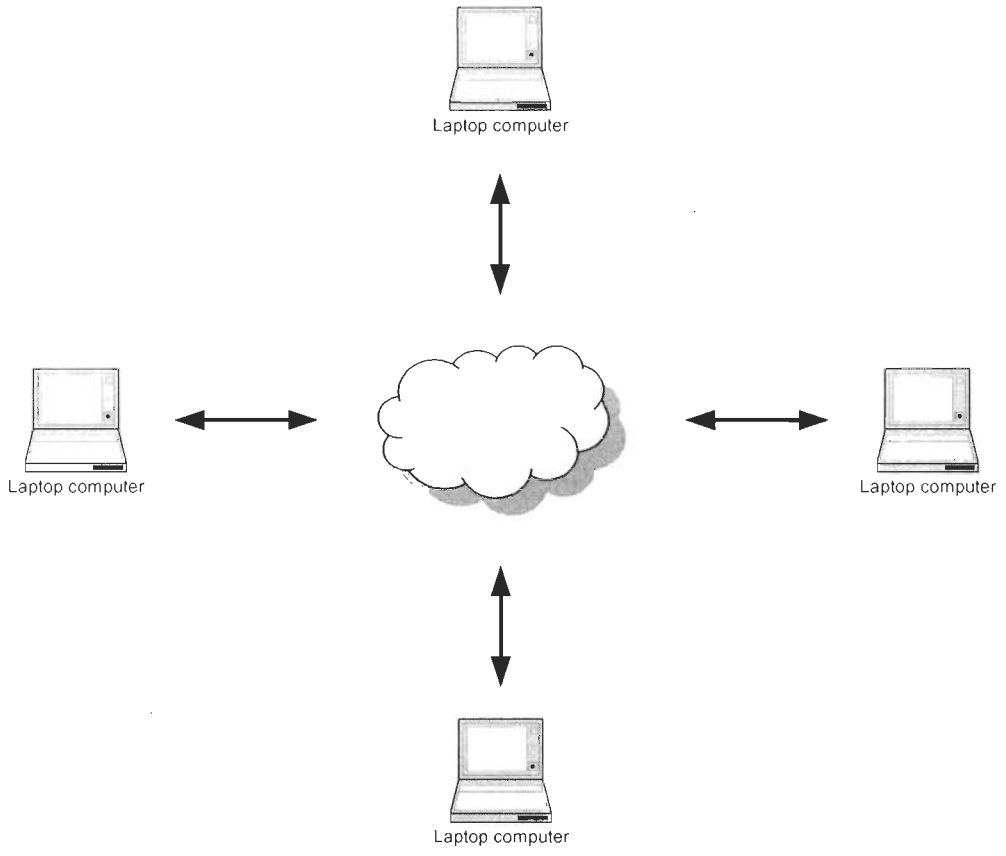


Figure 1.2: Many-to-many group communication

1.4 Group communication in wireless environment

In recent years the popularity of wireless network is also increasing very fast. Wireless technology is gaining more and more attention from both academia and industry. Most wireless networks used today, e.g. the cell phone networks and the 802.11 Wireless Local Area Networks (WLANs) [6,34,45,48,72,81,100,101], are based on the wireless network model with pre-existing wired network infrastructures. Packets from source wireless hosts are received by nearby base stations, then injected into the underlying network infrastructure and then finally transferred to destination hosts.

Another wireless network model, which is actively researched, is the mobile ad hoc network (MANET) [10,17,42,44,63,82]. This network is formed by only mobile nodes and requires no pre-existing network infrastructure, as shown in Fig 1.3. Nodes with wireless

capability form a MANET instantly and packets can be delivered to any node in the network. Since there is no base station and no underlying network infrastructure in MANETs, some mobile nodes work as routers to relay packets from source to destination. Mobile nodes that are within the communication range of each other can communicate directly whereas the nodes that are far apart have to rely on intermediary nodes (routers) to relay messages. The mobility of a node in the mobile ad hoc networks can cause frequent changes in the network topology. These networks also have limited bandwidth.

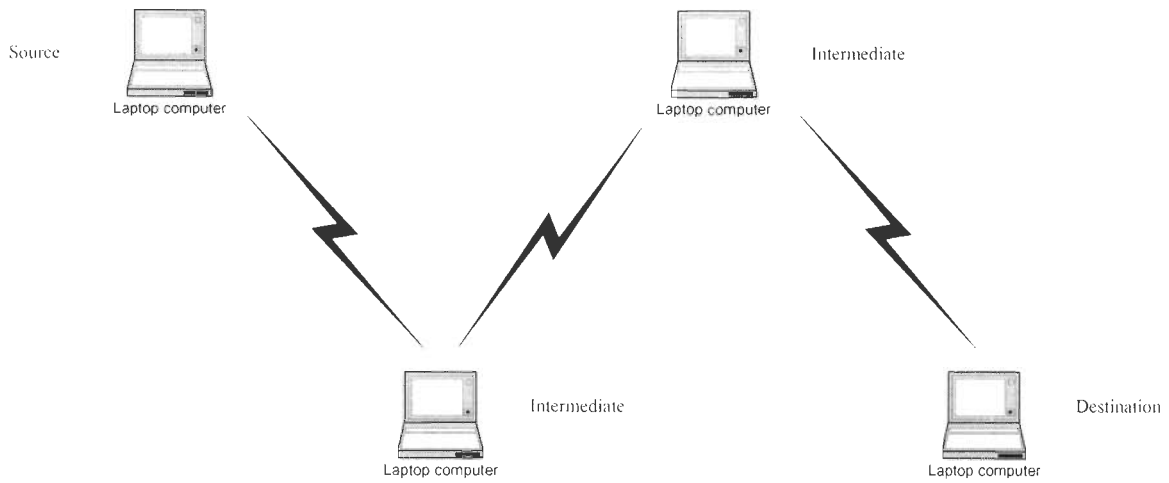


Figure 1.3: Mobile Ad Hoc Network (MANET)

It's very easy and economic to form an MANET in real time. MANET is ideal in situations like battlefield or emergency rescue area where fixed network infrastructure is very hard to deploy. The Table 1.1 depicts some applications of MANETs in different areas.

Many group-oriented network applications can be easily conducted in this new network environment. For example, in a conference room or in battlefield, users can form an ad-hoc network instantly with their wireless devices, e.g. notebook computers, PDAs, or even cell phones, without requiring any pre-installed cables or base stations. They can use this fast setup ad-hoc network for conducting a videoconference, sharing files or even playing interactive games. Before the ad-hoc network concept can be widely accepted, several issues need to be resolved [10,17,42,43,63,82]. For example, security and efficiency are major challenges in this environment.

Table 1.1: Applications of Mobile Ad Hoc Networks

Applications	Descriptions / Services
Tactical networks	Military communication, operations Automated Battlefields
Emergency services	Search-and-rescue operations as well as disaster recovery; e.g., early retrieval and transmission of patient data (record, status, diagnosis) from / to the hospital. Replacement of a fixed infrastructure in case of earthquakes, hurricanes, fire, etc.
Commercial environments	E-Commerce, e.g., electronic payments from anywhere (i.e., in a taxi). Business: Dynamic access to customer files stored in a central location on the fly provide consistent databases for all agents, Mobile office Vehicular Services: Transmission of news, road conditions, weather, music, Local ad hoc network with nearby vehicles for road / accident guidance.
Home and enterprise networking	Home / office wireless networking (WLAN), e.g., shared whiteboard application, use PDA to print anywhere, trade shows Personal area network (PAN)
Educational applications	Set up virtual classrooms or conference rooms
Entertainment	Multi-user games, Robotic pets, Outdoor Internet access
Location-aware services	Follow-on services, e.g., automatic call forwarding, transmission of the actual workspace to the current location Information services: Push, e.g., advertise location-specific service, like gas stations Pull, e.g., location-dependent travel guide; services (printer, fax, server, gas stations) availability information; intermediate results, state information, etc.

In order to specify what efficiency and security mean, one may consider some desirable properties for a group key establishment protocol. Efficiency, while not measuring security, is to be considered as a crucial property when designing key establishment protocol and it is

quantified as the number of communication rounds, number of messages, number of exponential operations required to create a group key. Secondly, security is another very important aspect of any group key establishment protocol. The general goal of a secure communication is to establish a common secret key (also referred to as a group key), among all group members for confidential communication. Usually, a secret group key is established by a group key establishment protocol. Therefore, each group key establishment protocol should satisfy certain security properties, viz., forward secrecy, backward secrecy, collusion freedom, and key independence.

Forward secrecy: It requires that members who left the group should not have access to any future key. This ensures that a member cannot decrypt data after it leaves the group. To assure forward secrecy, a renewal of group key through rekey operation is required after each leave operation.

Backward secrecy: It requires that a new member that joins the session should not have access to any old key. This ensures that a member cannot decrypt data sent before it joins the group. To assure backward secrecy, a renewal of group key through rekey operation is required after each join operation.

Collusion freedom: It requires that any set of fraudulent members should not be able to collude and deduce the current group key.

Key independence: A protocol is said key independent if a disclosure of a key does not compromise other keys.

Many group key establishment protocols have been proposed in the literature. Some protocols [1,3-5,8,9,12,13,15,16,18,19,21,22,27-33,35,36,38-41,46,50,53,57-62,65,66,68,71, 75-78,84-93,96,98,99,104,105,107,110,112-116] are designed for group key establishment in general, while others [7,37,51,55,56,70,73,80,83,95,103,108,111] are proposed for group key establishment problem in ad-hoc networks. Unfortunately, none of these protocols is fully adaptive to the key establishment problem in mobile ad-hoc networks. Some key establishment protocols among them depend on a reliable central key server, whereas others require a lot of rounds, messages, and exponentiation operations for group key establishment. A common problem of these protocols is that, by design, they did not take into account the unique features of mobile ad-hoc networks (MANETs), for example, less computational power and communication bandwidth etc. Key establishment protocols should be designed so that computational load is distributed among multiple nodes. Heavy computation work on a

single node should be avoided. Secondly, radio signal used in wireless communication propagate in every direction in space. So in wireless network, a local broadcast to all neighbors within the radio range uses no significant more time and resource than a unicast. A well-designed group key establishment protocol should take these features into consideration and should adapt to the mobile ad-hoc network environment.

1.5 Problem statement

Efficient, scalable, reliable, and secure communication services have become critical in modern computing. Many collaborative applications (e.g., conferencing, white-boards, shared instruments, and command-and-control systems) need secure communication. However, experience shows that security mechanisms for collaborative dynamic peer groups tend to be both expensive and unexpectedly complex. Collaborative dynamic peer groups are different from non-collaborative, centrally managed, one-to-many broadcast groups such as those encountered in Internet. Security requirements of collaborative dynamic peer groups, particularly in mobile Ad hoc networks (MANETs), present interesting research challenges. Key establishment and management, as the cornerstone of most other security services, presents the initial and formidable obstacle. Although centralized key management might initially appear attractive, it is inherently unsuitable for dynamic peer groups. The rationales are: first, centralization violates the peer nature of the group by concentrating all key generation in a single point; and secondly, a centralized key server becomes a single point of failure, an attractive attack target and heavily computation oriented. Of course, a key server can be sufficiently replicated and fortified to address some of these issues. However, it is very costly to guarantee the availability of a key server in unreliable ad hoc network, thus, each group member must be prepared to become a key server. Although centralized and decentralized group key establishment protocols work efficiently in wired networks yet these protocols may not work well in MANETs because of their peculiarities, i.e., nodes are lightweight devices like PDAs, Laptops, etc.. Moreover, nodes have limited battery power, wireless links between nodes form and dissolve unpredictably, and their bandwidths are limited as well. Given the stringent resources, group key establishment should be lightweight for the sake of efficiency to conserve bandwidths, energy, storage, and computations. The main objective of the present research work can be described as follows:

"To design and analyze an efficient protocol for group key distribution problem for mobile ad hoc network (MANET), and also provide a comparative study of proposed protocol against various existing group key establishment protocols".

In order to explore the above problem, it could be divided into small number of objectives, which are as follows:

(i) Investigation of group key establishment protocols;

(ii) Complexity analysis of group key agreement protocols in setup phase;

(iii) Complexity analysis of group key agreement protocols for dynamic membership events, i.e., join, leave, multiple join, and multiple leave;

(iv) To propose a secure and efficient group key agreement protocol, its security and complexity analysis and compare it with clique protocols (IKA.1 and IKA.2);

1.6 Contributions of the thesis

This work investigates the group key establishment protocols extensively and finds that the group key agreement protocols may provide better solution for mobile ad hoc networks because these are contributory wherein each member provides equal contribution in the creation of the group key. Major contributions of this thesis may be described as follows:

(i) Extensive investigation of group key establishment protocols for mobile ad hoc networks;

(ii) Complexity analysis of group key agreement protocols based on two party Diffie-Hellman protocol, where it is concluded that the Clique protocols (IKA.1 and IKA.2) show the best results;

(iii) Complexity analysis of group key agreement protocols, based on two party Diffie_Hellman protocol, for dynamic membership events, i.e., join, leave, multiple join and multiple leave, where it is again concluded that the Clique protocols (IKA.1 and IKA.2) show the best results

(iv) A secure and efficient group key agreement protocol is proposed, and its security and complexity analysis is done. The protocol is compared it with Clique

protocols (IKA.1 and IKA.2). It is observed that the proposed protocol shows better performance as compared to Clique protocols (IKA.1 and IKA.2);

1.7 Organization of thesis

The rest of the thesis is structured as follows: Chapter 2 presents the comprehensive investigation of group key establishment protocols. Chapter 3 presents the complexity analysis of group key agreement protocols based on two party Diffie-Hellman protocol for mobile ad hoc networks. Chapter 4 describes group key agreement protocols for dynamic membership events. Chapter 5 presents a new secure and efficient group key agreement protocol, and its security and complexity analysis. The comparison of the protocol with other two best protocols concluded in Chapter 3 and 4 is also presented. Lastly, Chapter 6 concludes the thesis.

Chapter 2

Literature review

2.1 Group Key Establishment Protocols

Although a slew of group key establishment protocols have been proposed in the literature, yet no major effort has been made to establish the suitability of these protocols to mobile ad hoc networks (MANETs). Efficient handling of events, such as join, leave, merge and partition, is a major issue for any group key establishment protocol. Some protocols, therefore, propose to organize the secure group into subgroups with independent local group keys. This reduces the impact of re-keying, but requires subgroup leaders, which are responsible for communication among subgroups. Other protocols suggest that a central entity create and distribute the group key. Therefore, we can broadly classify the existing solutions into two approaches: common group key approach [1,3-5,8,9,12,13,15,16,21,22,28-32,35,36,38-41,46,50,60-62,71,76-78,84,88,91,98,99,104-106,113-115] and the independent group key per sub-group approach [18,53,85,86,96,112,116] as depicted in Fig 2.1.

2.2 Notations

Following notations will be used in subsequent chapters.

ρ: the signature,

H: hash function,

$|p|$: the length of a prime number p where p is an order of a cyclic group G,

g: one-way function,

n: total number of members in the group,

j: the number of joining (merging) member(s),

l: the number of leaving (partitioning) member(s).

2.3 Common group key approach

In this approach, all group members share a common group key. The group key establishment architecture can be classified into two categories, viz, distributory and contributory architectures. Distributory architecture can be further categorized into centralized and decentralized architectures.

In centralized architecture only one central server is responsible for creation and distribution of keys. There are three centralized approaches, viz, pairwise keys approach, broadcast secrets approach and key hierarchy approach.

In decentralized architecture based on common group key approach, one central server creates and other additional servers help in distributing keys, this way other servers share the load of the central server. Decentralized architecture can be further classified into two approaches based on re-keying operation such as membership-driven re-keying and time-driven re-keying.

In contributory architecture each node contributes equally in group key creation, thereby reducing the amount of computation to be done by each node. The contributory architecture can be further classified into different categories such as ring-based, star-based, hierarchy-based, broadcast-based and secret share-based. The complete architectural chart is shown in Fig 2.1.

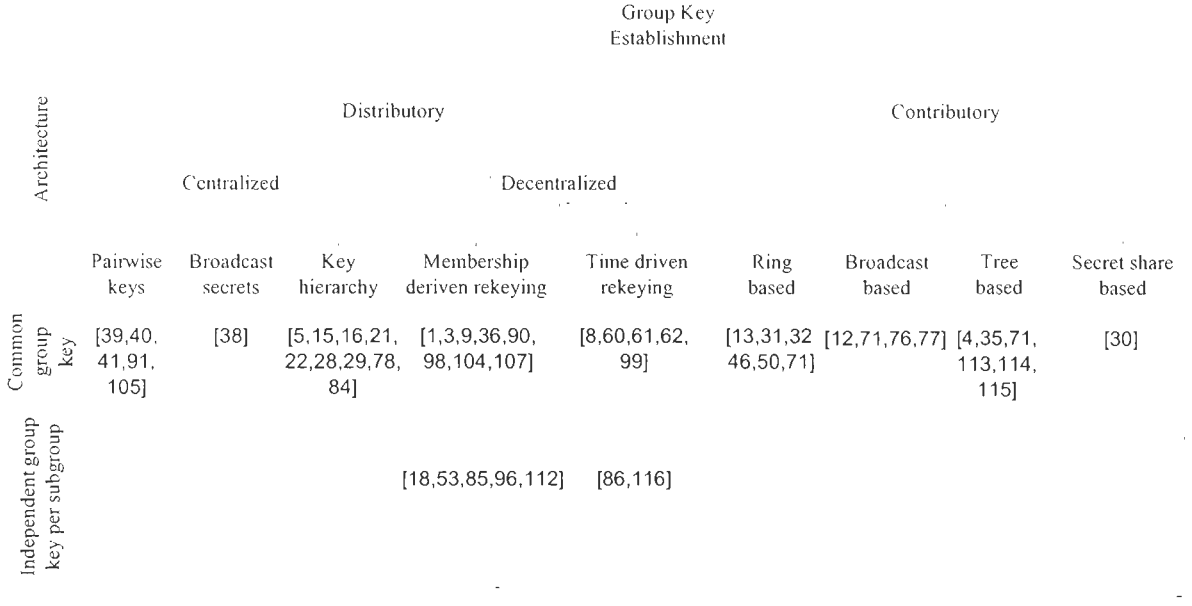


Figure 2.1: Group Key Establishment Protocols

2.3.1 Pairwise Keys Approach

In this subcategory of protocols [39,40,41,91,105], the key server shares a secret key with each group member. These keys are generally called key encryption keys and are used to establish secure channels between the key server and each member in order to re-distribute the current group key securely whenever required.

2.3.1.1 Group Key Management Protocol by Harney and Muckenhirn

Harney and Muckenhirn proposed the Group Key Management Protocol (*GKMP*) [39,40] based on pairwise keys approach. The *GKMP* proposes to create symmetric keys and distribute them amongst communicating peers. The *GKMP* assumes that there is a Group Key Management (*GKM*) application available and executable by any node in the network. In *GKMP,* the security manager plays the role of key distribution center and also authenticates and validates a new member's identity. Initially a group originator is identified for group, which first obtains a certification from a trusted entity verifying that the originator is responsible for generating and distributing the group key. The originator verifies each join request using the security manager before sending it to *GKM* application. Initial member list is used by the *GKM* application to generate the Group Key Packet (*GKP*). The *GKP* contains the current Group Traffic Encrypting Key (*GTEK*) and the future Group Key Encrypting Key (*GKEK*) .The *GKMP* allows *GKM* application to identify itself as the group key controller, which is validated by the member participating in the *GKP* generation. Once the *GKP* is generated, the group controller distributes it to every member of the group by contacting and verifying its security parameter. And also generates a Session Key Packet (*SKP*) for that member. The *SKP* contains traffic encryption key (*TEK)* and key encryption key (*KEK)* and has the form: *SKP* = (*STEK*, *SKEK)*. This *SKP* is the first given to the member encrypted using the public key of the member. The controller then uses the *SKEK* of that member to encrypt the *GKP* and then creates a Group Rekey Package (*GRP*) for that member. When the group has to be rekeyed, the *GKMP* allows *GKM* application to contact any member of the group and generate a new *GKP* and then broadcast it after encrypting using the old *GKEK*. This protocol has many disadvantages, for example, a group originator is responsible for generating and distributing the group key, each join request is validated by security manager and if a current member is compromised, the proposed method of rekeying will not be effective since the compromised member still has the access to the old *GKEK* and can

decrypt the new *GKP*. Since *GKMP* requires $O(n)$ rekey messages for each leave from the group, this solution does not scale to large groups with highly dynamic members.

2.3.1.2 Group key management protocol by Dunigan and Cao

Dunigan and Cao [105] proposed a protocol wherein architecture uses public/private keys to verify member identity. A group access policy defines what is required to join the group. A suite of cryptographic functions provides encryption, hashing, random number generators, and digital signatures. Any network host can create and manage a group and be designated as group authority, which specifies the group name, group id, the security policy and cryptographic algorithms required, and communication requirements (multicast address, port, time-to-live and management port). This information is conveyed to group members in group information token that is signed by the group authority. Using group information token, a member joins the group by communicating with the group controller. The group controller enforces the group policy, validating a member's id, signature and access certificates. The group controller manages key lifetimes and re-keying as well as compromise recovery. Group keys are managed by means of a simple message protocol between group controller and group member. A member wishing to join a group first obtains the group information either from a pre-delivered certificate or by sending a group discovery message to the group authority. The group join message is signed by the sender and includes the sender's Diffie-Hellman value and any certificates required by the group policy. The group controller processes the group join request by validating the signature on the requester's message and conforming the required certificates are valid and meet the group policy. If the requester is authorized, the controller generates his Diffie-Hellman shared secret key. The controller also generates the group key and sends back a group join reply signed with the controllers' key. The member receives the reply, verifies the signature, calculates the Diffie-Hellman shared secret key and uses it to decrypt the group key block. The group controller also monitors key lifetimes and multicasts a group re-key message, which contains a new *GTEK* encrypted under the *GKEK* key. If members notice their keys are out of date, they can send a key update request to the group controller asking for the current group key. The policy field in the group information token specifies the type of authorization required to join a group. Since group controller is responsible for managing group keys, enforcing the group policy, validating a member's id, signature and access certificates, monitoring key lifetimes,

keeping each member's network address and responsible for re-key or rejoin by sending a unicast message to each member, it is prone to different kind of failures.

2.3.1.3 Poovendram et al. protocol

Poovendram et al. [91] have also proposed a scheme similar to *GKMP* [39,40], where authentication and authorization functions are delegated to other group members rather than centralized at the same group controller entity. The paper addresses the two issues, i.e., robustness and scalability. In this protocol the notion of group controller is changed to a panel of controllers. This panel consists of three members at any given time. Among these three, one serves as the active group controller, with the group keys being generated by two panel members with the constraint that no two-panel members may participate in consecutive key generations. This approach allows every panel member to have only shared key generation authority. At least two members of the control panel have to agree in order to re-key the group. Replacing a single group controller with a panel of three members adds more functionality to the panel and reduces the probability of failure of the whole group controller panel. If one member is compromised, then the other two members can inform the security manager to effectively remove the compromised panel member. In case of if more than one panel member are compromised, the protection mechanism is at the same level as the *GKMP* or *KDC* in terms of the ability to deal with a compromised members.

It is noticed that authentication, verification, join-authorization, session key parameter negotiation and distribution have to be scalable for a dynamic multicast group. To effectively scale the key exchange, access control list *(ACL)*, revocation control list *(RCL)*, and other issues, authors propose to have the group first segmented into clusters and each cluster managed by a sub-controller panel of three members which has all the authorities of the main control panel except the group key packet *(GKP)* generation.

In this scheme, only one panel is identified as core and is responsible for generating the group key packet, which includes the *GTEK* and *GKEK*. The core panel then transmits this information to the cluster panels, which will then use the information in generating the session keys, etc. The size of any particular group is the deciding factor in formation of clusters and hence the sub-group panels. The clustering process involves cluster formation, cluster splitting, cluster merging, and cluster joining. Cluster formation involves setting up the cluster sub-control panel id, establishing a cluster id, and setting up the link to the

neighboring clusters, if any. Although this protocol eliminates the single point of failure by introducing a set of panel members in place of one group controller, yet these panel members can be compromised.

2.3.1.4 Hao-hua Chu et al. Protocol

Hao-hua Chu et al. [41] also proposed a solution for group key establishment based on pairwise keys approach. Their key distribution algorithm is designed to achieve the goals of security in open network environment, multicast architecture independence, and robust dynamic membership support. To start a new secure multicast group, key distribution algorithm requires only a group leader is to be started. Data transmission can be divided into three phases: sending phase, verification phase, and receiving phase. In sending phase, the sender constructs a data message that contains three components. The sender multicasts the data message in the multicast channel. When the data message is received from the multicast channel, members cannot decode the data yet because they do not have the message key. Only the group leader has symmetric key which can be used to decode message key. The verification phase involves the group leader. Upon receiving the data message from the sender, the group leader looks up its current membership list to check that the sender is indeed a valid group member. Then it decrypts the second and third component of the data message using its symmetric key with sender to obtain the message key. And verifies that the (member_id, message_id) pairs are the same between the first component and the second component in the data message. When both validation checks succeed, the group leader prepares a verification message, which is then signed with the private key of the group leader. The group leader signs and multicasts the valid verification message in the multicast channel. The last phase involves the receiver listening on the multicast channel. Upon receiving a data message, the receiver separates the data message into three components. The receiver decrypts it with the public key of the group leader to reveal the message tag. If the verification message contains the *VALID* symbol, the receiver uses his/her assigned *uid* and searches for his/her slot that contains the message key in the verification message. After the message key is decrypted, the receiver uses the message tag to retrieve the corresponding encrypted data component from the data queue. Then the receiver can decrypt the data with the message key. This protocol has the drawback to require the transmission of the validation multicast message by the group leader, with a size in the order of $O(n)$ (n being the number of

current valid group members), after each time the source sends a message to the group. Another drawback is group leader, which has the authority and the necessary information to accept / reject new membership requests. Its' failure may result the whole group inoperative.

2.3.2 Broadcast Secrets Approach

In this subcategory of protocols, the rekeying of the group is based on broadcast messages instead of peer-to-peer secret transmissions.

2.3.2.1 Secure Locks Protocol

Chiou and Chen [38] proposed Secure Lock key management protocol where the key server requires only a single broadcast to establish the group key or to rekey the entire group in case of a leave. The protocol relies on Chinese remainder theorem. In this protocol, the key server assigns a positive integer, m_i, to each member, and shares a secret value k_i with each of them. When the server wants to send a message to the group, it generates a random value K and uses it to encrypt the message. Then, it encrypts K with each secret k_i and obtains the set $\{K_i\}$ of the encryptions of $K(K_i = \{K\}_{k_i})$. Then the server computes a lock M. Then, Server multicast lock M as well as the encrypted message with K. Upon reception of the lock M, each member recovers the encryption key $K = \{M \bmod m_i\}_{k_i}$, and hence decrypts the received message. Members, whose secret k_i and its corresponding positive integer m_i are included in the computation of the lock M, can recover the decryption key K. This protocol minimizes the number of re-key messages. However, its drawback is the increase in computation load at the server due to the Chinese remainder calculations before sending each message to the group. And, server is also prone to different kind of attacks.

2.3.3 Hierarchy of Keys Approach

In pair-wise keys approach, re-keying operations require high number of update messages (in the order of $O(n)$). The key server establishes a secure channel individually with each member and uses this channel to distribute the group key updates. In order to reduce the number of update messages, in this class of protocols, the key server shares secret keys with subgroups of the entire secure group in addition to the individual channels. Then, when a member leaves the group session, the key server uses the secret subgroup keys that

are unknown to the leaving member, to distribute the new group key. Thereby, subgroup secret keys help in reducing the required number of update messages. In subsequent sections, we present some protocols that use this concept.

2.3.3.1 Logical Key Hierarchy (LKH) Protocol

Wong et al. [15,16,28] and Wallner et al. [29] proposed the Logical Key Hierarchy (LKH) protocol. In LKH, the key server maintains a key tree. The nodes of the tree correspond to key encryption keys and the leaves of the tree correspond to secret keys shared with the members of the group. Each member holds a copy of its leaf secret key and all the key encryption keys corresponding to the nodes in the path from its leaf to the root. The key corresponding to the root of the tree is the group key. For a balanced binary tree, each member stores at most $(\log_2 n+1)$ keys, where n is the number of group members. The key hierarchy assists in reducing number of re-key messages from $O(n)$ to $O(\log_2 n)$, for example, for a group of six members $(u_1,u_2,u_3,u_4,u_5,u_6)$, the key server builds a hierarchy of keys as shown in Fig 2.2.

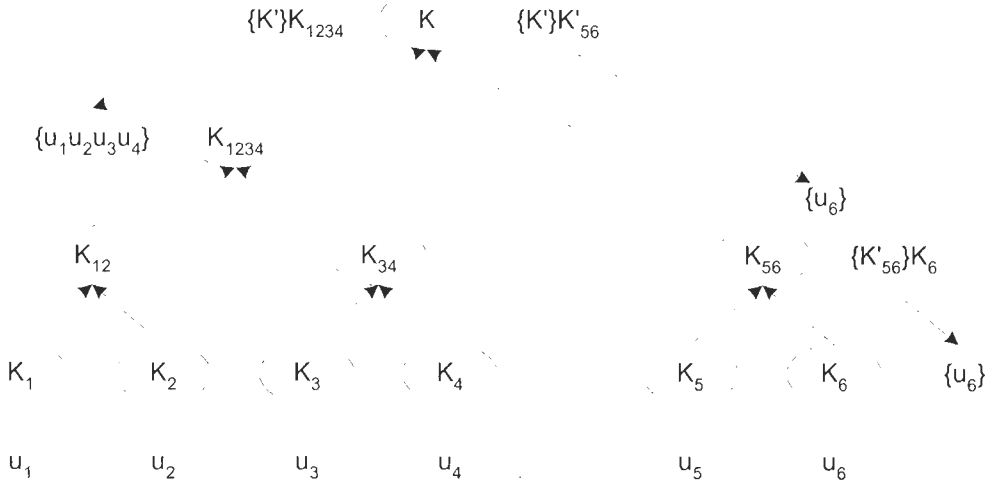


Figure 2.2: Logical key hierarchy approach

Each member owns a secret key, which is a leaf in the tree as well as all the keys on its path to the root. The root represents the group key K shared by all the members. The other keys are used to reduce the required re-keying messages User u_1 owns (K_1,K_{12},K_{1234},K), u_2 owns (K_2,K_{12},K_{1234},K), u_3 owns (K_3,K_{34},K_{1234},K), u_4 owns (K_4,K_{34},K_{1234},K), u_5 owns (K_5,K_{56},K) and u_6 owns (K_6,K_{56},K). Let us assume that u_5 leaves the group, key

server updates K_{56} into $K_{56}^{'}$, sends $K_{56}^{'}$ to u_6 encrypted with K_6. Group key K is updated into K' and sent to (u_1, u_2, u_3, u_4) encrypted with K_{1234} and to u_6 encrypted with $K_{56}^{'}$ and hence only three messages are required instead of five messages if GKMP were used.

Wong et al. [15] proposed the extension of the binary key tree to a k-ary key tree. Using a greater degree reduces the number of keys maintained by the members because of a smaller tree depth. Performance analysis shows that optimal results are reached with trees having a degree less or equal to 4. This protocol has many drawbacks, for example, the key server is responsible for creating the key tree, which makes it vulnerable to attacks and failures. Another drawback is that each member stores the key tree, which needs a lot of memory. This protocol also needs a lot of re-key messages whenever join or leave takes place.

2.3.3.2 One-way Function Trees (OFT) Protocol

McGrew and Sherman [21,22] proposed an improvement over LKH called One-way Function Trees (OFT). OFT assists in reduction of the number of re-key messages from $2\log_2 n$ to only $\log_2 n$. With OFT, a key encryption key is calculated by members rather than attributed by the key server. Indeed, each key encryption key K_i is computed using its child key encryption keys using the formula:

$$K_i = f(g(K_{left(i)}), g(K_{right(i)}))$$

where *left*(i) and *right*(i) denote the left and right children of node i respectively, and g is a one-way function. The result of applying g to a key K: $g(K)$ is called the blinded key version of K.

In this protocol, each member maintains its leaf secret key and its blinded sibling key and the set of blinded sibling key encryption keys of its ancestors. Fig 2.3 illustrates the ancestors and their corresponding sibling keys of member u_2. Using aforementioned formula, each member can calculate all the required ancestor key encryption keys (key encryption keys on the nodes in the path from the leaf secret to the root) recursively. In the original scheme (LKH), when a new key encryption key is generated, it is encrypted with its two child key encryption keys. However, in OFT when a blinded key is changed in a node it has to be encrypted only with the key of its sibling node. Therefore, the required number of re-key messages is reduced to half.

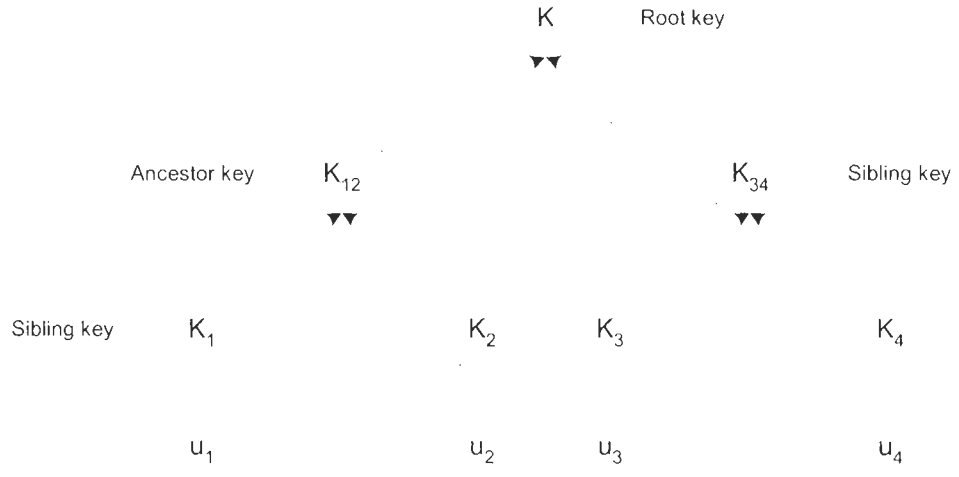


Figure 2.3: Ancestor and sibling keys of member U_2

2.3.3.3 Canetti et al. Protocol

Canetti et al. [84] proposed a approach, which reduced communication overhead to half of Wallner et al. [29] approach. The proposed scheme is called one-way function chain tree that uses a pseudo-random generator to generate the new key encryption keys rather than a one-way function. The initialization of the scheme is the same as in [29]. The user revocation procedure is as follows: Let G be Pseudo-random generator, which doubles the size of its input [74]. $L(x)$ and $R(x)$ denote the left and right halves of the output, $G(x)$, i.e., $G(x) = L(x)R(x)$ where $|L(x)| = |R(x)| = |x|$. To remove a user u, the group controller associates a value r_v to every node v along the path from u to the root as follows: It chooses $r_{p(u)} = r$ at random and sets $r_{p(v)} = R(r_v) = R^{|u|-|v|}(r)$ for all other v (where $p(v)$ denotes the parent of v). The new keys are defined by $k'_v = L(r_v) = L(R^{|u|-|v|-1}(r))$. Notice that from r_v, one can easily compute all keys $k'_v, k'_{p(v)}, k'_{p(p(v))}$ up to the root key k'. Finally each value $r_{p(v)}$ is encrypted with key $k_{s(v)}$ (where $s(v)$ denotes the sibling of v) and sent to all users. For example, in order to remove user u_0 from the tree of Fig 2.4, we send encryptions $E_{k_{001}}(r), E_{k_{011}}(R(r)), E_{k_1}(R(R(r))$. Each user can compute from the encryptions all and only the keys it is entitled to receive. This construction halves the communication overhead of the basic scheme to only $\log n$, and its security can be rigorously proven. It has an additional

advantage that group key is the output of Pseudo-random generator whereas in [29], it is chosen by group controller.

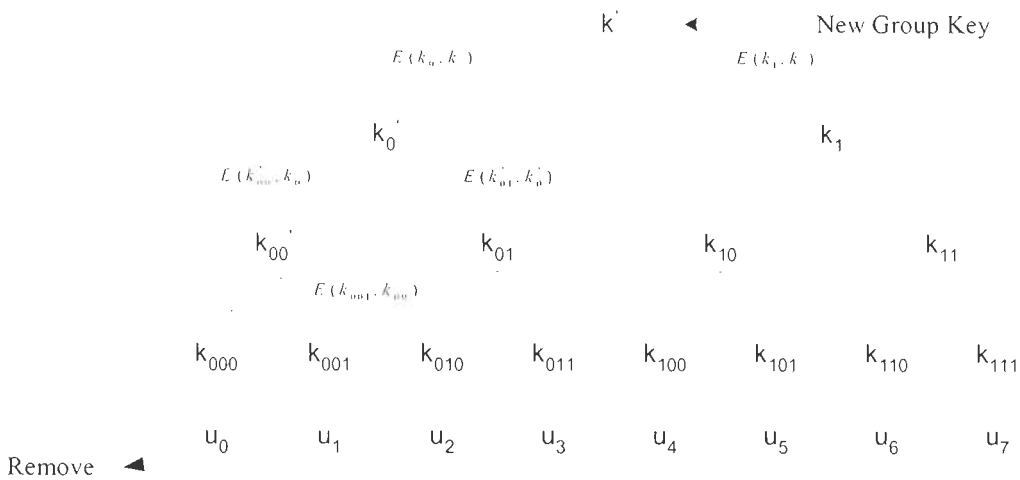


Figure 2.4: Key revocation in the basic scheme

This protocol also has same disadvantage as LKH protocol [15] and OFT protocol [29], that is, group controller, whose failure may render the whole protocol inoperative.

2.3.3.4 Efficient Large-Group Key Distribution (ELK) Protocol

Perrig et al. [5] proposed Efficient Large group Key distribution (ELK) protocol, which uses pseudo random functions to generate the new key encryption keys. There is a family of Pseudo-Random Functions (PRFs) that uses key K on input M of length m bits. A number of derivation functions are also used. In ELK all members are at leaves in the logical key hierarchy, ELK is composed of two basic mechanisms: Key update and Key recovery (through hints). In case of leave event, ELK uses a new key update protocol, where the left and right child keys contribute to update the parent key. This approach is similar to the OFT protocol [22], but this paper allows small hints that allow legitimate members to reconstruct the key without the key update. In key recovery, instead of broadcasting the key update message, legitimate members who know K and either K_L or K_R can also recover the new key K' from a hint that is smaller than the key update message, by trading off computation for communication. The paper assumes that a member can perform 2^m computations that construct a smaller key update or a hint. The group key distribution protocol allow users to join and leave the group at any time or in aggregate join and leave requests that occur in one time interval into one group key update, which makes key update message smaller.

Member join event: In a member join event, the key server assigns the new member to a node in the key tree and the new member receives all the updated keys on the path from its leaf node to the root which involves higher communication overhead. The ELK trade off computation for lower communication overhead.

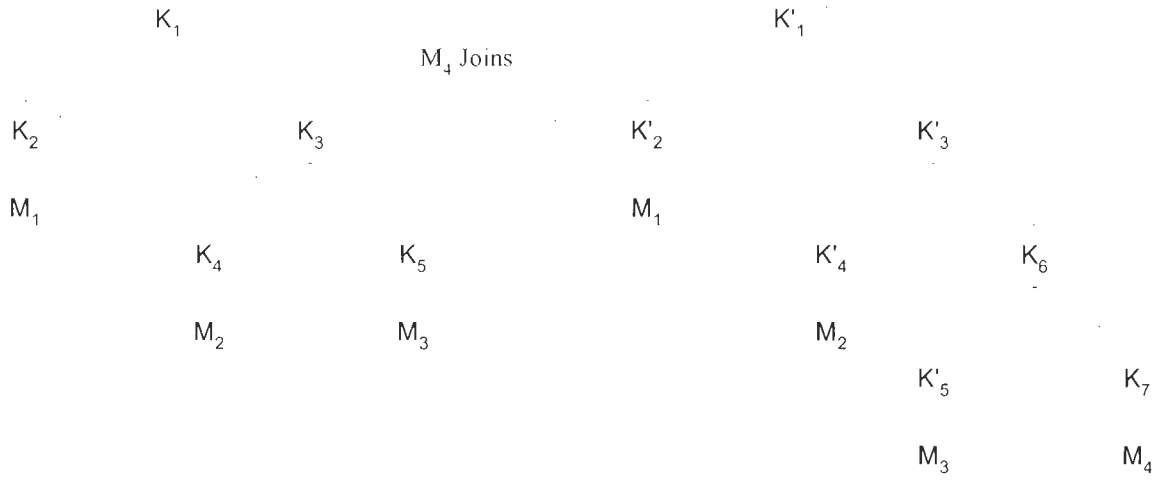


Figure 2.5: Member join event

Fig 2.5 shows an example join event where member, M_4, joins the group and the server decides to insert it at the leaf node of M_3. In the first step, the server updates all keys in the tree. Step 2 does not apply since no empty leaf nodes are available. In step 3 the key server generates the new leaf node and assigns it a new random key K_7. In step 4, the server decides to merge M_4 to the node of $K_5^{'}$, and generates the parent node M_6 of $K_5^{'}$ and K_7. The server computes the new key, K_6. In step 5 the server sends M_4 the message $\{K_1^{'}, K_3^{'}, K_6\}_{K_7}$. In step 6, the server sends the joining location of the new member to M_3 by unicast, which tells M_3 to update its key tree and to compute K_6. In case of multiple members join events the members can be placed in the empty locations this way no overhead is generated. If no empty leaf nodes exist, the key server can first generate a smaller key tree with the new members, and join that tree to one node of the current group tree. In this case the server only needs to communicate the location of a single node to the members that live below the joining node.

Member leave event: The member leave event is more complicated than the member join event, because all the keys that the leaving member knows need to be replaced with new keys that the leaving member must not be able to compute to ensure forward secrecy. The key

server uses the child contribution scheme discussed in single join event to update the keys on the path from the leaf node of the leaving member up to the root.

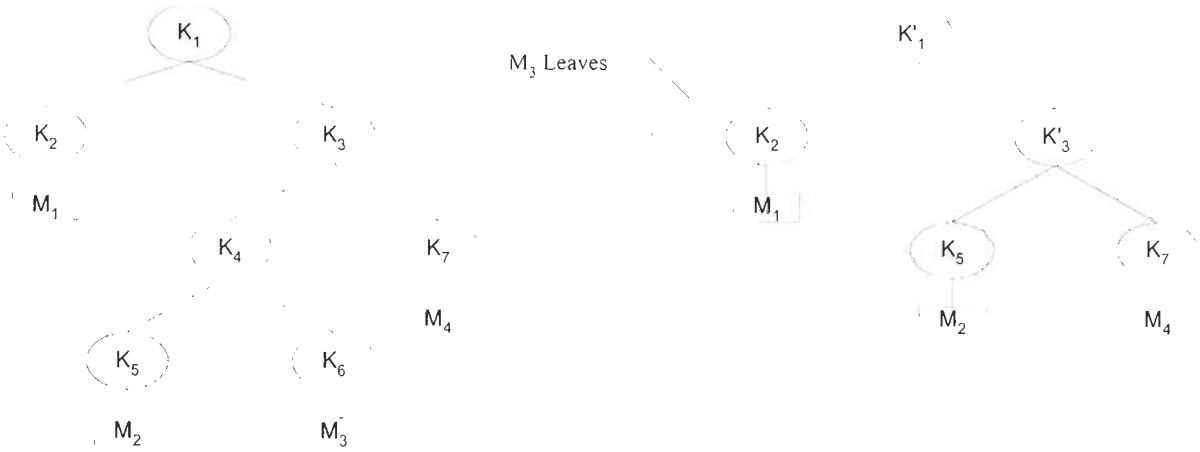


Figure 2.6: Member leave event

Fig 2.6 shows the leave event where member M_3 leaves the group. In step 1, the key server deletes the nodes that correspond to the keys K_4 and K_6. The server promotes the node of K_5. In step 2, the server updates the keys on the key path of the leaving member K_3 and K_1. In step 3 the server broadcasts the key update. Finally, subsequent data packets contain the hint. In case of multiple members leave in the same interval, the key server aggregates all the leaving members and creates a joint leave key update message. ELK can aggregate the concurrent member leave events particularly well and provides (in addition to current savings) a 50% saving over OFT protocol [22] for keys when both children change. The reason for OFT's inefficiency is that if both child keys change, OFT needs two key updates (one for each child), whereas ELK only needs one.

2.3.3.5 Centralized Tree-Based Key Management (CTKM) by Waldvogal et al.

Waldvogel et al. [78] proposed and implemented a centralized, easy maintainable scheme, which achieves tightest control over the individual participants. It suits to applications with high security demands, and poses very little load on the network and the receivers. The group manager centrally manages all keying material. To store the keying material, a tree is used in which all participating entities are represented by its leaves. Fig 2.7 depicts the hierarchy of keys. During setup phase, which includes admission control, each participant establishes a shared secret with the group manager. The group manager stores it in

the leaf node associated with this participant, and uses it whenever only this individual participant should understand a message such as for unicast traffic during this participants join operation. Its revision is increased after each use to ensure perfect forward secrecy. Besides incrementing the revision field, the keying material is passed through a one-way function so that the newcomer cannot recover earlier traffic. The node x in the binary tree held by the group manager contain further KEKs, used to achieve efficient communication of new keying material when the membership of the group changes.

Each participant holds a different subset of keys from the tree, more specifically all those keys that are in the path from the participants leaf to the root node, which is used as the TEK. These intermediate KEKs are used if a message should only be understood by a part of the group, e.g., a message encrypted with KEK 47 is understood by participants 4...7. This enables the transmission of new keys to only a limited set of participants, thereby disabling others to decrypt specific messages.

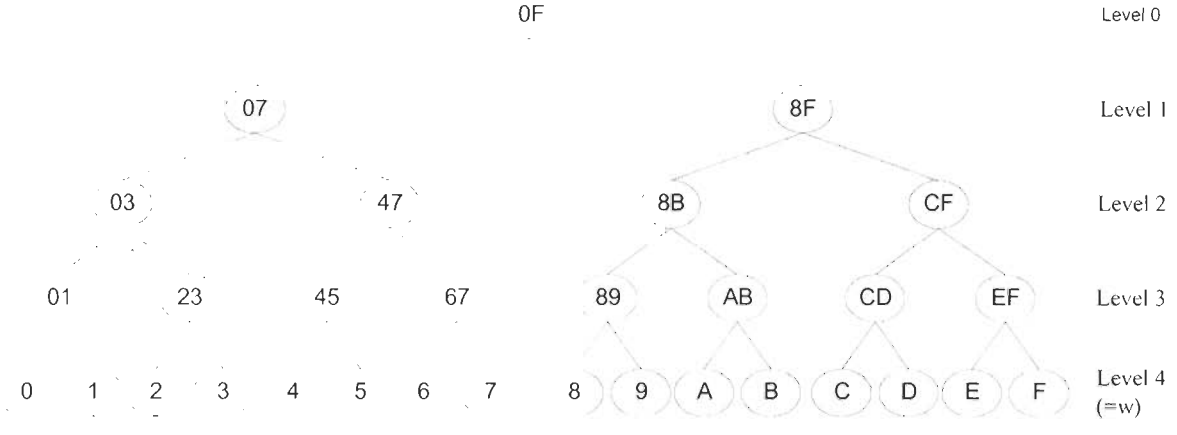


Figure 2.7: Binary hierarchy of keys

Single Join/ Multiple join: On a join operation, the participant's Key Manager unicasts its request to the group manager, which checks with admission control and assign an ID (say 4), where the participant's individual key is stored. The ID is used such that the bit-pattern of the ID defines the traversal of the tree, leading to a unique leaf. Participant's ID may be his IP address and port number, or a function thereof. The group manager increases the revision of all the keys along the path from the new leaf to the root (KEKs 45, 47, 07 and TEK 0F), puts them through the one-way function and sends the new revision of the keys to the joining participant, together with their associated version and revision numbers. At the same time, all senders are informed of the revision changes, so they start using the new TEK. If several

joins happen in short succession, the revision of the TEK and the KEKs shared between the newcomers only need to be increased once, if newcomers can be allowed to decipher a small amount of data sent out before they were admitted (usually only a fraction of a second).

Single Leave/ Multiple leave: To perform a leave operation, the group manager sends out a message with new keying material, which can only be decrypted by all remaining participant's key managers. Additionally, it frees the slot utilized by the leaving participant, making it available for reuse at a future join. Assume C is leaving. This means that the keys it knew (KEKs CD, CF, 8F, and TEK 0F) need to be viewed as compromised and have to be changed in such a way that C cannot acquire the new keys. This is done efficiently by following the tree from the leaf node corresponding to the leaving participant to the TEK stored in the root node, and encrypting the new node keys with all appropriate underlying node or leaf keys. Using a single message for multiple leaves take advantage of path overlaps, so several keys will only be created and sent out once per message instead of once per leave operation. This can be used to efficiently coalesce multiple leave (and join) operations into a single message.

2.3.3.6 Centralized Flat table Key Management (CFKM) protocol by Waldvogel et al.

Waldvogel et al. [78] proposed the Centralized Flat Table Key Management protocol (CFKM). In this approach, the key hierarchy is changed into flat table in order to reduce the number of keys maintained by the Key Server. The Table 2.1 contains a single entry for the traffic encryption key (TEK) and 2w entries for the key encryption keys (KEKs), where w is the number of bits in a member identifier (the authors propose to use IP addresses as member identifiers). Two keys are associated to the two possible values of each bit in a member id. Table 2.1 illustrates the structure for w = 4. Each member holds the key encryption keys (KEKs) associated to the values of its identifier bits. Thus, each member holds w +1 keys (w key encryption keys in addition to a traffic encryption key). For instance, a member with the identifier 0101 maintains KEK00, KEK11, KEK20, KEK31 and a traffic encryption key (TEK). To join, a participant contacts the key server, where it is assigned a unique ID and receives the keys corresponding to the ID's bit/value pairs. For example, a newcomer with (binary) ID 0010 would receive the TEK and the key encryption keys (KEKs) K30, K20, K11, and K00 over the secure setup channel, after their revision was increased. When a member leaves the group, all the keys held by this departing member should be modified to

assure forward secrecy. Therefore, the key server sends a re-key message containing two parts: a first part contains the traffic encryption key (TEK) encrypted with each not compromised key encryption key (KEK) from the flat table, and hence all the remaining members would be able to decrypt the new traffic encryption key (TEK_{new}).

Table 2.1: Centralized flat table for w=4

	TEK	
id bit #0	*KEK*00	*KEK*01
id bit #1	*KEK*10	*KEK*11
id bit #2	*KEK*20	*KEK*21
id bit #3	*KEK*30	*KEK*31
	bit 0	bit 1

The second part contains the new KEKs encrypted with both the old KEK and the new traffic encryption key (TEK_{new}). This way, the leaving member cannot recover the new traffic encryption key (TEK_{new}) and the remaining members can update their old KEKs without having access to the KEKs of other members.

Table 2.2: CFKM Rekey messages when member 0101 leaves the group

	TEK	
id bit #0	$\{\{KEK00_{new}\}KEK00_{old}\}TEK_{new}$	$\{TEK_{new}\}KEK01$
id bit #1	$\{TEK_{new}\}KEK10$	$\{\{KEK11_{new}\}KEK11_{old}\}TEK_{new}$
id bit #2	$\{\{KEK20_{new}\}KEK20_{old}\}TEK_{new}$	$\{TEK_{new}\}KEK21$
id bit #3	$\{TEK_{new}\}KEK30$	$\{\{KEK31_{new}\}KEK31_{old}\}TEK_{new}$
	bit 0	bit 1

Table 2.2 illustrates the re-key message sent by the key server after the leave of the member with the identifier 0101. This protocol has many disadvantages, for example, the key server, which maintains the keys, is prone to failures. And, members have to maintain a number of key encryption keys (KEKs) and traffic encryption key (TKE), which requires a lot of memory.

2.3.4 Membership Driven Re-keying

In this subcategory of protocols, the group key is changed whenever a join or a leave operation occurs in the group. Some protocols based on this approach are as follows:

2.3.4.1 Scalable Multicast Key Distribution

Ballardie [1] proposed the scalable multicast key distribution (SMKD) protocol, which uses the tree built by the Core Based Tree (CBT) multicast routing protocol [3,104] to deliver keys to multicast group members. In the CBT architecture, the multicast tree is rooted at a main core. Secondary cores can exist eventually. The main core creates an access control list (ACL). Group key and key encryption key (KEK) are used to update the group key. The ACL, the group key and the key encryption key are transmitted to secondary cores and other nodes when they join the multicast tree after their authentication. Any router or secondary core authenticated with the primary core can authenticate joining members and use the ACL to distribute the keys, but only the main core generates those keys. One disadvantage with SMKD is that there is no provision to handle the forward secrecy problem. Hence, in case of leave operation whole group is created afresh.

2.3.4.2 Intra-domain Group Key Management Protocol

DeCleene et al. [9] and Hardjono et al. [107] proposed the Intra-domain Group Key Management Protocol (IGKMP). This architecture divides the network into administratively scoped areas. There are a Domain Key Distributor (DKD) and many Area Key Distributors (AKDs). Each AKD is responsible for distributing the group key to members within one area. Fig 2.8 exemplifies this architecture. The group key is generated by the DKD and is propagated to the group members through the AKDs. Because distribution of group key within the area must be secure, area-local keys are used by AKD to distribute a new group key to members within the area. The DKD and AKDs belong to a multicast group called All-KD-Group. The DKD uses this group to transmit re-key messages to the AKDs who re-key in turn their respective areas. The drawback of this architecture is that it suffers from a single point of failure, which is the DKD, the entity responsible for generating the group key. Besides, in case of an AKD failure, members belonging to the same area will be not able to access the group communication.

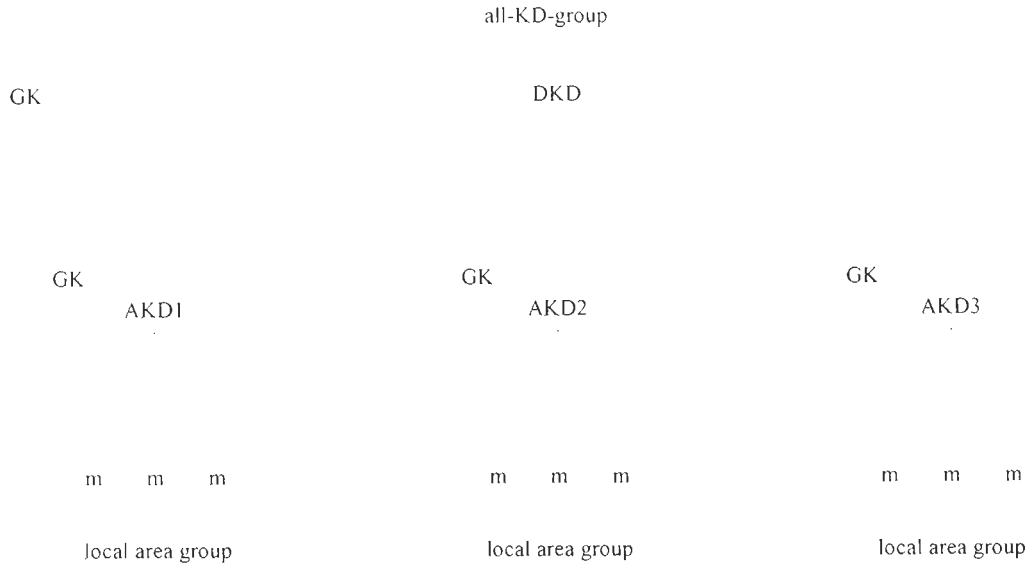


Figure 2.8: Intra-domain Group Key Management Protocol architecture

2.3.4.3 Hydra Protocol

Rafeli and Hutchison [98] proposed Hydra protocol wherein the group is organized into smaller subgroups and a server called the Hydra server (HS_i) controls each subgroup *i*.

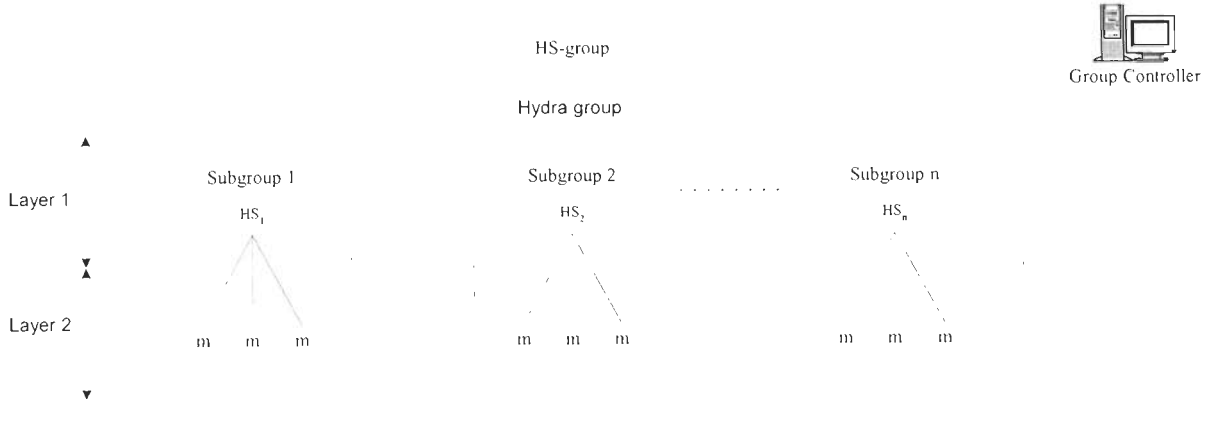


Figure 2.9: Hydra architecture

If a membership change occurs at subgroup *i*, the corresponding HS_i generates the group key and sends it to the other HS_j involved in that session. In order to have the same group key distributed to all HSs, a special protocol is used to ensure that only a single valid HS is generating the new group key whenever required. Fig 2.9 depicts the Hydra architecture.

2.3.4.4 Baal Protocol

Baal Chaddoud et al. [36] proposed a decentralized group key establishment protocol called Baal, which is similar to Hydra protocol [98]. This protocol mainly addresses the problem of scalability. The protocol defines three entities such as Group Controller (GC), which maintains a participant list and creates and distributes the group key to group members via local controllers, Local Controllers (LC) which manage the keys within its subnet. When a LC receives a new group key, it distributes it to the members connected to its subnet. Besides, a LC can play the role of the GC by generating and distributing new group keys after membership changes following some coordination rules, and Group member who belongs to participation list.

When a membership change occurs at a subnet, the corresponding LC can generate a new group key and distribute it to its subnet and to the other members via their LCs. To assure that a single LC generates a new group key at a time, the GC assigns a priority to each LC and when many LCs distribute simultaneously a new group key, the LCs are instructed to commit to the group key issued by the LC having the highest priority.

This protocol suffers from many problems such as it has group controller and local controllers, which are responsible for creating and distributing group key. In case of failure of these entities either the whole protocol will fail or part of the group communication system become inoperative.

2.3.5 Time-Driven Re-keying

In this subcategory of protocols, the group key is changed after each specific period of time. Thereby, the departing members are not excluded immediately from having access to the secure content. Similarly, new members are delayed up to the beginning of a new interval of time. This is a major drawback of these protocols wherein join and leave events take place frequently. Some protocols available in the literature, which are based on this approach, are as follows:

2.3.5.1 Kronos Protocol

Setia et al. [99] proposed the Kronos protocol, which is driven by periodic re-keying rather than membership changes that means a new group key is generated after each time interval rather than after each membership change. In Kronos protocol, each domain is

divided into many areas managed by different AKDs as in IGKMP protocol. However, in Kronos the DKD does not multicast the group key each time to the AKDs. Instead of that, each AKD generates independently the same group key whenever required and re-keys the members belonging to its area. To implement this scheme, the AKDs' clocks should be synchronized, and the AKDs have to agree on a re-key period. Second, the DKD transmits secret factors K and GK_0 to AKDs using secure channels. To generate the group key GK_{i+1}, AKDs calculate after each period of time: $GK_{i+1} = E_K(GK_i)$, which is the encryption of the previous group key (GK_i) with the encryption algorithm E using the secret key K.

2.3.5.2 MARKS Protocol

Briscoe [8] proposed the MARKS protocol wherein slicing of time length is suggested. And, each time slice uses a different key for encrypting the data. The encryption keys are leaves in a binary hash tree that is generated from a single seed as shown in Fig.2.10.

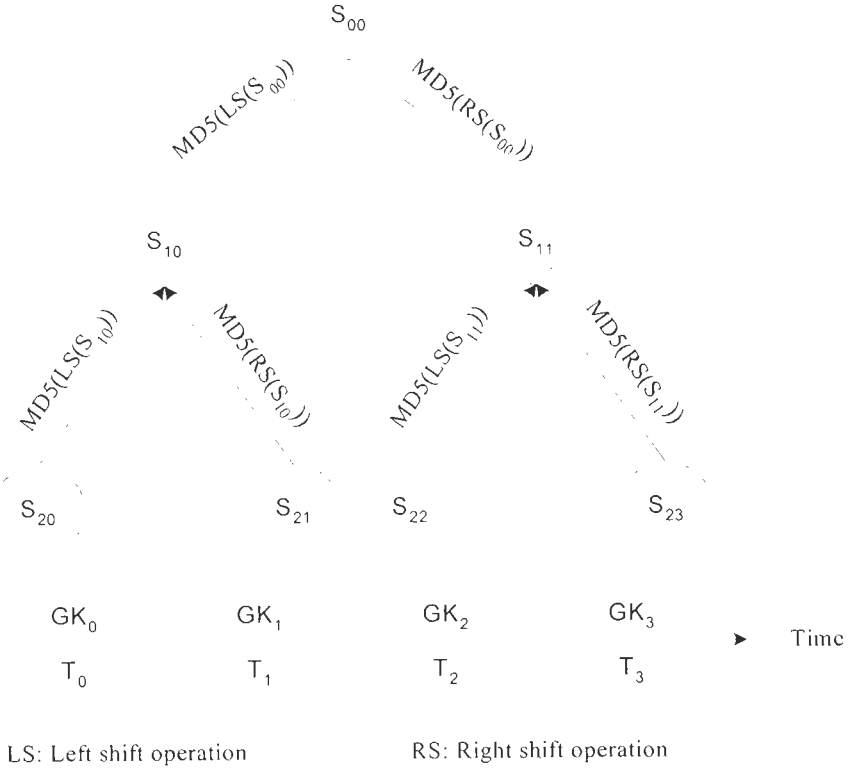


Figure 2.10: MARKS Key Generation Tree

A blinding function, such as MD5 [92] is used to create the tree nodes. Fig 2.10 depicts an example of the generated binary tree whose leaves are the keys that correspond to the different slices. Each intermediate node (including the root) is allowed to generate two children (left and right children). The left node is generated by shifting its' parent one bit to the left and applying the blinding function on it. The right node is generated by shifting its' parent one bit to the right and applying the blinding function on it. Users willing to access the group communication receive the seeds needed to generate the required keys. The system cannot be used in situations where a membership change requires the change of the group key, since the keys are changed as a function of the time. The distribution of the seeds and the management of receivers' queries are assured by a set of key managers.

2.3.5.3 Dual Encryption Protocol (DEP)

Dondeti et al. [60-62]] proposed the Dual Encryption Protocol (DEP) to eliminate the problem of trusting third parties in decentralized protocols. They suggest hierarchical subgrouping of the group members where a sub-group manager (SGM) controls each subgroup. There are three type of KEKs and one Data Encryption Key (DEK), KEK_{i1} is shared between a SGM_i and its subgroup members, KEK_{i2} is shared between the Key Server (KS), and the group members of subgroup i excluding SGM_i. Finally, KS shares KEK_{i3} with SGM_i. In order to distribute the DEK to the group members, the KS generates and transmits a package containing the DEK encrypted with KEK_{i2} and encrypted again with KEK_{i3}. Upon receiving the package, SGM_i decrypts its part of the message using KEK_{i3} and recovers the DEK encrypted with its subgroup KEK (KEK_{i2}), which is not known by the SGM_i. SGM_i encrypts this encrypted DEK using KEK_{i1} shared with its subgroup members and sends it out to subgroup i. Each member of subgroup i decrypt the message using KEK_{i1} and then, decrypting the message using KEK_{i2} (shared with KS) and receives DEK. Therefore, the DEK cannot be recovered by any entity that does not know both keys. Hence, although there are third parties involved in the management (SGMs), they do not have access to the group key (DEK). When the membership of subgroup i changes, the SGM_i changes KEK_{i1} and sends it to its members. Future DEK changes cannot be accessed by members of subgroup i that did not received the new KEK_{i1}.

2.3.6 Ring-based cooperation

In this subcategory the nodes are logically arranged in the ring form. The following subsections describe some protocols available in the literature, which are based on this approach.

2.3.6.1 Ingemarsson et al. Protocol

Ingemarsson et al. [46] proposed INGM protocol. This is the first natural extension of two party Diffie-Hellman key agreement protocol [110]. The brief description of protocol is as follows: Let $G = \langle \alpha \rangle$ be a multiplicative group of some large prime order q. Assume that n participants $M_1, ..., M_n$ want to agree upon a common key. The participants must be arranged in a logical ring form. In a given round, every participant raises the previously received intermediate key value to the power of its exponent and forwards the result to the next participant. After $n-1$ rounds, all users agree upon a common group key $K = \alpha^{r_1 r_2 r_3 ..... r_n}$. The INGM protocol is an extension of two party Diffie-Hellman protocol [110]. This protocol has the advantage that each node contributes equally in group key formation. Thereby, this protocol may provide a solution of group key establishment problem in mobile ad hoc network.

2.3.6.2 Burmester and Desmedt (cyclic) Protocol

Burmester and Desmedt [71] presented this protocol. This is similar to the broadcast system except that a bi-directional cyclic network is used. So $M_1, ..., M_n$ are linked in a cycle, with M_i connected to M_{i+1}. In this protocol, each user M_i computes $z_i = \alpha^{r_i} \bmod p$ and sends it to both users M_{i-1} and M_{i+1}. If any user fails in bi-directional cyclic network, the whole protocol will stop, that is why, this protocol is not suitable for mobile ad hoc networks.

2.3.6.3 Burmester and Desmedt (Star-based) Protocol

Burmester and Desmedt presented this protocol in [71]. The $M_1, ..., M_n$ be a dynamic set of users who want to generate a group key wherein user, M_1, is the chair. Each user, M_i, computes $z_i = \alpha^{r_i} \bmod p$. Then M_1 sends z_1 to all M_i and the M_i sends z_i to M_1. Due to its design this protocol is not suitable for mobile ad hoc networks.

2.3.6.4 Boyd and Neito's Group Key Agreement Protocol

Boyd and Neito [13] proposed a single-round authenticated static group key agreement that meets the bound of Becker and Wille [12] for a single round protocol and have proved the security in the random oracle model [67] following the model established by Bellare et al. [65,66,67]. The protocol does not provide forward secrecy, which is a major drawback of this protocol because in dynamic environment the leaving member should not get access to future communication among group members.

2.3.6.5 Bresson, Chevassut and Pointcheval's group key agreement protocol

Bresson et al. [31] provided a formal treatment of the authenticated group Diffie-Hellman key exchange problem in a scenario in which the membership is dynamic rather than static.

The authenticated dynamic group key agreement scheme consists of three protocols setup, remove and join. Suppose a multi-cast group of users $I = \{U_1, ..., U_n\}$ wish to agree upon a common key. They are arranged in a ring. Each user saves the set of values he receives in the down-flow of setup, remove and join operations. Any user from I could be selected as a group controller U_{GC} trusted to initialize the dynamic operations. In this protocol, authors consider the user with the highest index in I as the group controller, the flows of a user U are signed using its long-lived key, LL_U, and the session key SK is:

$$SK = H(I \mid FI_{\max(I)} \mid g^{x_1 \cdots x_{\max(I)}})$$

where $FI_{\max(I)}$ is the down-flow, session identities SIDS and partner identities PIDS are appropriately defined. Drawback of this protocol is that it uses a group controller for initialing the dynamic events, which are not possible when the group controller is not working.

2.3.6.6 Bresson, Chevassut, Essiari and Pointcheval's group key agreement protocol

A very efficient provably secure group key agreement is introduced in [32] for dynamic scenario, which suits for restricted power devices and wireless environments. The scheme consists of three protocols: the setup protocol GKE.Setup, the remove protocol GKE.Remove and the join protocol GKE.Join. The main GKE.Setup protocol allows a cluster of mobile users (also called clients) and a wireless gateway (also called server) to agree on a session key. The other two protocols of the scheme handle efficiently the dynamic membership

changes in one wireless domain. The protocol executes in two rounds. In the first round, server, S, collects contributions from individual clients and then, in the second round it sends the group keying material to the clients. The protocols GKE. setup and GKE.join perform their operation in six steps each. However, GKE.remove protocol takes only three steps. This protocol has one disadvantage. In first round, the server, S, collects contributions from individual clients and then, in the second round it sends the group keying material to the clients. Therefore, in case of server failure, the protocol will not be workable.

2.3.6.7 Nam, Kim, Kim and Won's group key agreement protocol

Nam et al. [50] presented a communication-efficient dynamic group key agreement protocol well suited for a lossy and high-delay unbalanced network environment consisting of mobile hosts with restricted computational resources and stationary hosts with relatively high computational capabilities. They analyzed their scheme in the random oracle model [67] and proved that it is secure under factoring assumption. The scheme consists of three algorithms IKA1, LP1 and JP1 for initial group formation, user leave and user join respectively.

This protocol provides a solution for lossy and high-delay unbalanced network environment consisting of mobile hosts with restricted computational resources and stationary hosts with relatively high computational capabilities, which is not suitable for mobile ad hoc network wherein required nodes are computationally less capable and have limited memory.

2.3.7 Broadcast-based cooperation

In this subcategory the nodes use broadcast cooperation for creating the group key. The following subsections describe some protocols available in the literature based on this approach.

2.3.7.1 Burmester and Desmedt (broadcast-based) Protocol

Burmester and Desmedt [71] presented a much more efficient key agreement protocol in a group setting that requires only three rounds. When n users $M_1,.....,M_n$ want to agree upon a common conference key, they proceed as follows where the indices are taken modulo n so that user M_0 is M_n and user M_{n+1} is M_1.

Step 1: Each user M_i, $i = 1,.....,n,$, chooses a random $r_i \in Z_q$ and then computes and broadcasts $z_i = \alpha^{r_i} \bmod p$.

Step 2: Each user M_i, $i = 1,.....,n,$ checks that $ord(\alpha) = q$. Then it computes and broadcasts $X_i = (z_{i+1}/z_{i-1})^{r_i} (\text{mod } p)$.

Step 3: Each user M_i, $i = 1,.....,n,$ computes their session key as $K_i = (z_{i-1})^{nr_i} X_i^{n-1} X_{i+1}^{n-2} .....X_{i-2} (\text{mod } p)$

If all the users M_i, $i = 1,.....,n,$ follow the above steps, they will agree upon the same key $K = \alpha^{r_1 r_2 + r_2 r_3 + ...+ r_n r_1} (\text{mod } p)$.

The honest user computes the same key as, $K = \alpha^{r_1 r_2 + r_2 r_3 + ...+ r_n r_1} (\text{mod } p)$. The group key is computed as: Assume $A_{i-1} \equiv (z_{i-1})^{r_i} \equiv \alpha^{r_{i-1} \cdot r_i} (\text{mod } p)$, $A_i \equiv (z_{i-1})^{r_i} . X_i \equiv \alpha^{r_i \cdot r_{i+1}} (\text{mod } p)$, $A_{i+1} \equiv (z_{i-1})^{r_i} . X_i . X_{i+1} \equiv \alpha^{r_{i+1} \cdot r_{i+2}} (\text{mod } p)$, etc and we have $K_i = A_{i-1} . A_i . A_{i+1} .....A_{i-2}$. So the key is a second order cyclic function of the r_i. For $n = 2$ we get $X_1, X_2 = 1$ and $K \equiv \alpha^{r_1 r_2 + r_2 r_1} \equiv \alpha^{2 r_1 r_2} (\text{mod } p)$, which is essentially the same as for the Diffie-Hellman [110] system.

The BD (broadcast) protocol is an extension of two party Diffie-Hellman protocol [110]. This protocol has one advantage that it requires few number of rounds for it completion, therefore, it may be a prospective group key agreement protocol for mobile ad hoc networks.

2.3.7.2 The Octopus and Hypercube Protocols

Becker and Willie attempted to study lower bounds for the communication complexity of contributory key distribution in [12]. Their objective was to minimize the number of exchanges and for this they introduced the basic octopus protocol without broadcasting, which requires $2n - 4$ exchanges. They have described the hypercube or 2^d – cube protocol with d rounds and developed 2^d – octopus protocols with $\lceil \log_2 n \rceil + 1$ rounds that make use of the basic octopus protocol. Both these protocols use no broadcasting. Let G be a finite cyclic group of some large prime order q and α be a generator of G. We further assume a bijective mapping $\phi : G \to Z_q^*$. Suppose the participants $M_1, M_2 .....M_n$ want to agree on a common key.

Octopus Protocol: Before presenting this protocol, first consider the Diffie-Hellman key exchange among four users A, B, C, D as shown in Fig. 2.11. Users A and B and users C and D perform a Diffie-Hellman key exchange generating keys α^{ab} and α^{cd} respectively. Subsequently, $A(B)$ sends $\alpha^{\phi(\alpha^{ab})}$ to $C(D)$ and $C(D)$ $\alpha^{\phi(\alpha^{cd})}$ to $A(B)$. Hence, A and C (B and D) can generate the joint key $\alpha^{\phi(\alpha^{ab})\phi(\alpha^{cd})}$.

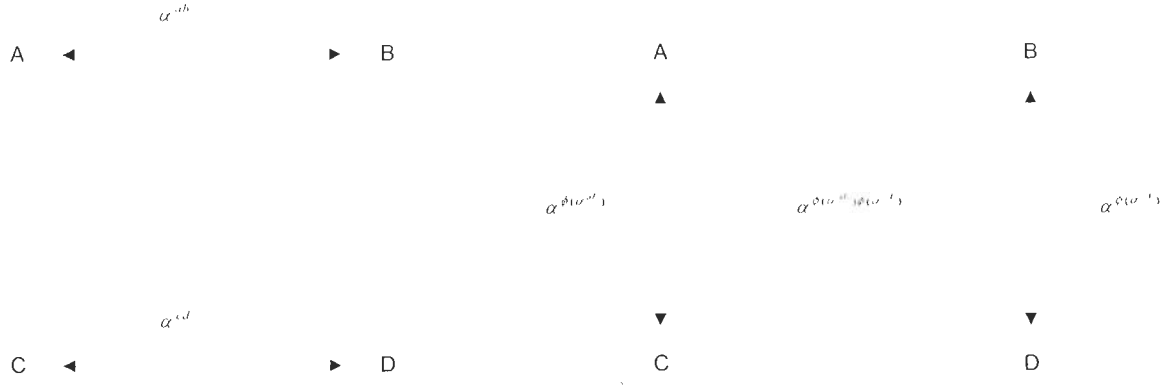


Figure 2.11: Diffie-Hellman Key exchange among 4 users

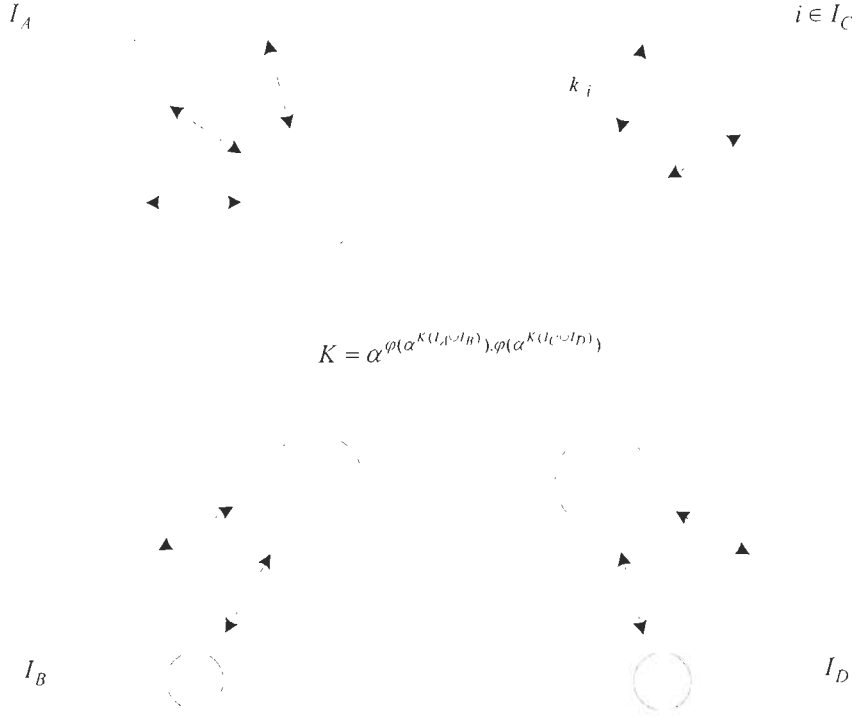


Figure 2.12: Octopus protocol

In the Octopus protocol, the participants $M_1, M_2 \ldots\ldots M_n$ are partitioned into five groups as shown in Fig.2.12. Four users $M_{n-3}, M_{n-2}, M_{n-1}$ and M_n take charge of the central control. We denote these users by A, B, C, D respectively. The remaining users are distributed in four groups: $\{M_i \mid i \in I_A\}$, $\{M_i \mid i \in I_B\}$, $\{M_i \mid i \in I_C\}$ and $\{M_i \mid i \in I_D\}$, where I_A, I_B, I_C, I_D are possibly of equal size, pairwise disjoint and $I_A \cup I_B \cup I_C \cup I_D = \{1, 2, \ldots\ldots, n-4\}$. Now, $M_1, M_2 \ldots\ldots M_n$ can generate a group key as follows:

Each user $X \in \{A, B, C, D\}$ generates a joint key k_i with user M_i for all $i \in I_X$. The users A, B, C, D perform the four party key exchange described above using the respective secret value $a = K(I_A)$, $b = K(I_B)$, $c = K(I_C)$ and $d = K(I_D)$, where $K(J) := \prod_{i \in J} \phi(k_i)$ for $J \subseteq \{1, 2, \ldots\ldots, n-4\}$. Thereafter, A, B, C, D hold the joint and later group key $K := \alpha^{\phi(\alpha^{K(I_A \cup I_B)})\phi(\alpha^{K(I_C \cup I_D)})}$. We describe this step for user A. The users B, C, D act correspondingly, for all $j \in I_A$, A sends $\alpha^{K(I_B \cup I_A \setminus \{j\})}$ and $\alpha^{\phi(\alpha^{K(I_C \cup I_D)})}$ to M_j. Now, M_j calculates $\alpha^{K(I_B \cup I_A \setminus \{j\})^{\phi(k_j)}} = \alpha^{K(I_A \cup I_B)}$ and then generates the group key $K = (\alpha^{\phi(\alpha^{K(I_C \cup I_D)})})^{\phi(\alpha^{K(I_A \cup I_B)})}$.

The Hypercube or 2^d – Cube Protocol: In the Hypercube protocol for 2^d participants, the 2^d participants are identified with the vectors in the d – dimensional vector space $GF(2)^d$ and a basis $\vec{b}_1, \ldots\ldots, \vec{b}_d$ of $GF(2)^d$ is chosen. The protocol may be performed in d rounds as follows: In the first round, every participant $\vec{v} \in GF(2)^d$ generates a random number $r_{\vec{v}}$ and performs a Diffie-Hellman key exchange with participants $\vec{v} + \vec{b}_1$ using the values $r_{\vec{v}}$ and $r_{\vec{v}+\vec{b}_1}$. In the i^{th} round, every participant $\vec{v} \in GF(2)^d$ performs a Diffie-Hellman key exchange with participant $\vec{v} + \vec{b}_i$, where both parties use the value generated in round $i-1$ as the secret value for the key exchange. In every round i, $1 \le i \le d$, the participants communicate on a maximum number of parallel edges of the d – dimensional cube in the direction $\vec{b}_i$, thus every party is involved in exactly one Diffie-Hellman exchange per round. Furthermore, all the parties share a common key at the end of this protocol because the vectors $\vec{b}_1, \ldots\ldots, \vec{b}_d$ form a basis of the vector space $GF(2)^d$.

2^d - **Octopus Protocol:** In order to formulate a protocol for an arbitrary number of participants ($\neq 2^d$), which requires a low number of simple rounds, the idea of the basic Octopus protocol is adopted in this protocol. In the 2^d – octopus protocol the participants act as in the octopus protocol introduced above with the only difference that 2^d instead of four parties are distinguished to take charge of the central control, whereas the remaining $n-2^d$ parties divide into 2^d groups. In other words, in steps 1 and 3 of the octopus protocol, 2^d participants manage communication with the rest and in step 2 these 2^d parties perform the cube protocol for 2^d participants. If the number of participants is n and if d is the largest integer smaller than $\log_2 n$, then the 2^d – octopus protocol requires $1+d+1=\lceil \log_2 n \rceil + 1$ simple rounds.

The 2^d – octopus protocol provides a tradeoff option between the total number of messages or exchanges needed and the number of rounds. For $d=2$ (Octopus protocol), the number of exchanges is optimal, whereas the number of simple rounds is comparatively high. On the other hand, if d satisfies $2^{d-1} < n \leq 2^d$, the number of simple rounds required is very low and the total number of messages is high. Furthermore, the protocol enables the group to decide how many participants should share control of the protocol.

The hypercube and octopus protocols are extension of two party Diffie-Hellman protocol [110], which require few number of rounds, messages, and exponential operations, therefore, these protocols may be suitable for mobile ad hoc networks.

2.3.7.3 Steiner, Tsudik and Waidner's Group Key Agreement Protocols

Steiner et al. [76] defined a class of generic n-party DH protocols such as GDH.1, GDH.2, and GDH.3. Let $G=\langle g \rangle$ be a cyclic group of a large prime order q. And, users $M_1, M_2, \ldots\ldots, M_n$ wish to agree upon a common session key.

GDH.1 Protocol: The protocol executes in $2(n-1)$ rounds and consists of two stages: upflow and downflow as shown in Fig. 2.13 and Fig. 2.14 respectively. The purpose of upflow stage is to collect contributions from all group members.

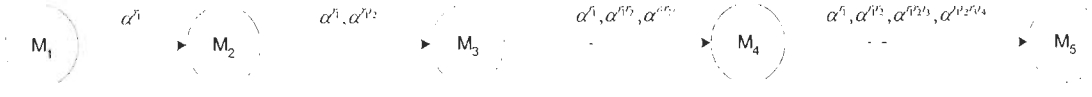


Figure 2.13: Upflow stage for 5 users in GDH.1 protocol

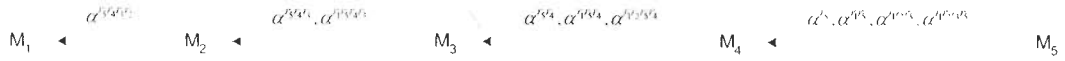


Figure 2.14: Downflow stage for 5 users in GDH.1 protocol

If all the users follow the above steps, they will agree upon a common secret key $K = \alpha^{r_1 r_2 \cdots r_n}$.

GDH.2 Protocol: The protocol executes in n rounds and consists of two stages. In the first stage ($n-1$ rounds) contributions are collected from individual group members and then in the second stage (n^{th} round), the group keying material is broadcasted. This is shown in Fig. 2.15. After two steps each user agrees upon a common secret key $K = \alpha^{r_1 r_2 \cdots r_n}$.

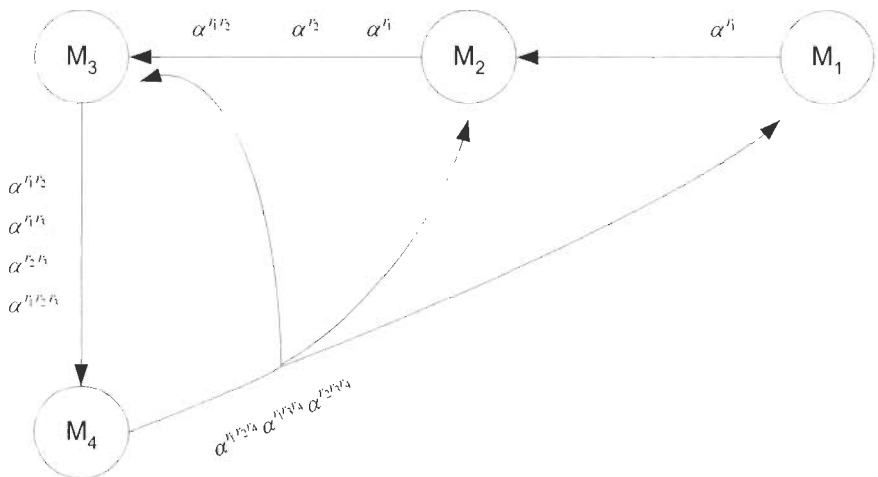


Figure 2.15: Stage 1 and 2 of GDH.2 protocol

GDH.3 Protocol: This protocol consists of four stages. The protocol execution among n users $M_1, M_2 \ldots\ldots M_n$ requires $n+1$ rounds and after step four each user M_i has a value of the form $\{\alpha^{\prod\{r_k | k\in[1,n]\wedge k\neq i\}}\}$ and can easily generate the intended group key $K = g^{r_1 r_2 \cdots r_n}$.

Note, Steiner et al. [76] extended GDH.2 and GDH.3 protocols for dynamic operations such as join, leave, partition and merge events and thereafter these protocols named as Initial Key Agreement.1 (IKA.1) and Initial Key Agreement.2 (IKA.2) respectively and complete protocols' suite is known as Clique protocol suite [77].

The IKA.1 and IKA.2 protocols are extension of two party Diffie-Hellman protocol [110]. The IKA.1 and IKA.2 protocols require few number of rounds, messages, and exponential operations, therefore, these protocols may be suitable for mobile ad hoc networks.

2.3.8 Tree-based Cooperation

In this subcategory the nodes use tree-based cooperation for creating the group key. The following subsections briefly describe some protocols available in the literature based on this approach.

2.3.8.1 Burmester and Desmedt (tree-based) Protocol

This protocol is similar to the star based system of BD protocol [71], except that a tree configuration network is used. The users $M_1,.....,M_n$ are labeled in such a way that the sons of M_a are M_{2a} and M_{2a+1}. The user, M_1, is the root of the tree which if fails then whole protocol becomes inoperable, that is why, it is not suitable for mobile ad hoc networks.

2.3.8.2 The TGDH Protocol

Wallner et al. first presented the binary tree-based approach for key establishment in [29]. Wong et al. then extended it for multiple degree tree in [15]. Yongdae Kim et al. proposed Tree-Based Group Diffie-Hellman (TGDH) protocol in [113,115]. TGDH combines a binary tree structure with the group Diffie-Hellman technique. They used one-way functions to enhance the security in protocol. Each node x in the key tree has two cryptographic keys, a secret key k_x and a blinded key $b_x = g(k_x)$, i.e., the blinded key b_x is computed from the secret key k_x using the one-way function g. The key is blinded in the sense that an adversary with limited computational capability can know b_x but cannot find k_x. The TGDH protocol uses the hierarchy in a binary tree to its advantage. The root is at the topmost level, given a value of *0* and all the leaves are at the lowest level *h*. Since the key tree is a binary tree, therefore, each node is either a leaf or a parent of two nodes. Each leaf node in the tree represents a group member M_i. The internal nodes are used for the key management and do not represent any individual member. Each node of the tree is represented by (l,v), where l is its level in the tree and v is the index of this node in level l.

The key associated to node (l,v) is $k_{(l,v)}$ and its blinded key $b_{(l,v)} = \alpha^{k_{(l,v)}} \bmod p$. Each group member contributes equally to the group key. In other words, for each internal node (l,v), its associated key $k_{(l,v)}$ is derived from its children's keys. In [113], they defined $k_{(l,v)} = b_{(l+1,2v+1)}^{k_{(l+1,2v)}}$, which is essentially $\alpha^{k_{(l+1,2v+1)} \cdot k_{(l+1,2v)}}$. The group key is a result of contributions by the current members. The finally key at the root (0,0) is calculated and subsequently hashed that results in group key. The TGDH protocol is an extension of two party Diffie-Hellman protocol, It uses the tree based approach for group key establishment problem. And, it provides a better solution for group key establishment when we measure it in terms of number of rounds, number of messages, and number of exponential operations. Therefore, this protocol may provide a solution to group key establishment problem in mobile ad hoc networks.

2.3.8.3 Collaborative Key Management Protocols by Adrian Perrig

Adrian Perrig presented three collaborative key management protocols in [4]. The design of protocols depends on the failure model as described by Powell [27]. There are two basic failure models: the crash failure or fail-stop model and the arbitrary failure model also known as Byzantine failure model. The arbitrary failure model describes the worst-case, where a program can behave in an arbitrary way [57,75,93]. The protocols described in the paper are based on fail-stop model, where members fail by halting. This is a reasonable model for practical application domains. The protocols in this paper assume that all members know the structure of the key tree and the position of each member in the tree.

The notations N, $E_k(M)$, $D_k(C)$, $H(M)$, $S_A(M)$, $S_A(M)$, $A \rightarrow B : M$, and $A \rightarrow * : M$ are number of members that are in the group or that wish to join it initially, encryption of plaintext M with key k, decryption of ciphertext C with key k, hash of message M, sign message M with $A's$ key; A sends message M to B, A broadcasts message M respectively.

Fig 2.16 shows the key tree of depth two. The root is at level 0 and lowest leaves are at level d. The nodes are denoted as $< l, v >$, where l stands for the level and i for the node number in the level, where $1 \le i \le 2^l$ since each level l hosts at most 2^d nodes. For each node $< l, v >$, there is a corresponding key $K_{<l,v>}$. The members are always placed at a leaf node.

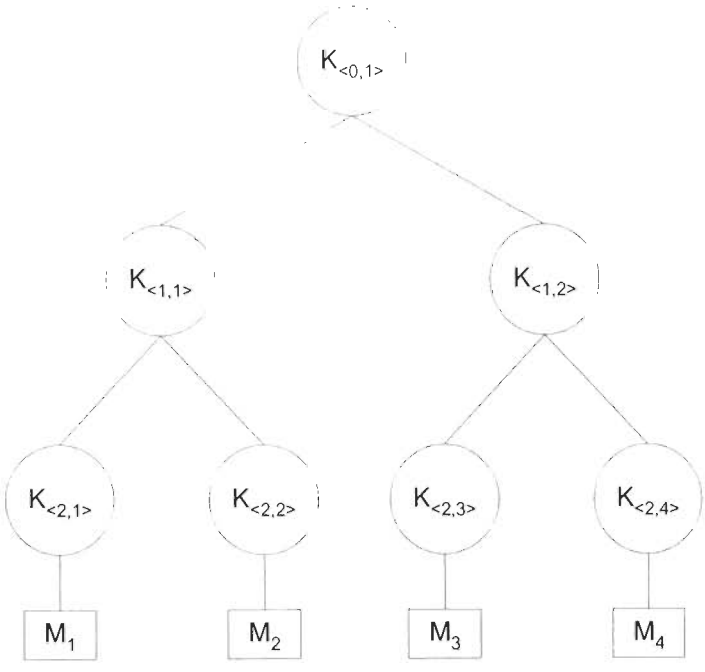


Figure 2.16: Notations of the nodes of a group key tree of depth = 2

Non-authenticated Group Key Agreement Protocol (NAGKA): This protocol is an extension of two party Diffie-Hellman protocol [110], and does not support authentication. However, it is useful in certain environment where the group members prefer to remain anonymous that is why it is called non-authenticated group key agreement (NAGKA) protocol. The key tree is constructed from the leaves up to the root as shown in Fig. 2.17. Initially each member chooses a random number, for the key value of its leaf node, which collectively forms the lowest level of the key tree. For the construction of each subsequent layer, two members establish a key through the key agreement procedure, one member from the left subtree and the other from the right subtree. They both broadcast their part of the Diffie-Hellman key agreement, which allows all the members of the subtree to compute the key of the current level. Once the group key is formed, the further communication is encrypted using this key.

Authenticated Group Key Agreement (AGKA): The AGKA protocol is similar to protocol NAGKA, except that two group members (which establish the parent key node of their current key nodes) also perform a mutual authentication when exchanging the exponents. The authentication is based on digital signatures and a public key infrastructure (PKI). When exchanging the keys, both members sign the concatenation of the exponent, the members' position in the tree, and a timestamp.

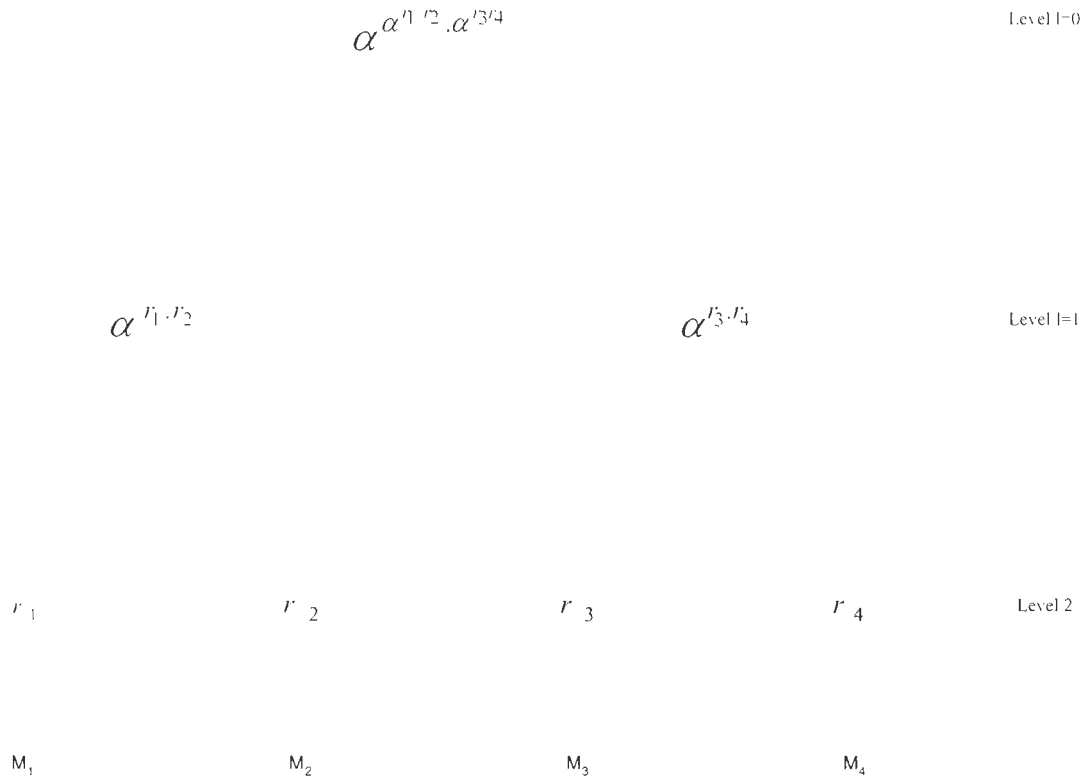


Figure 2.17: Non-authenticated group key tree (depth = 2)

Authenticated Group Key Agreement using Gunther's scheme (AGKA-G)

This protocol is similar to AGKA. Only difference is that it uses Gunther's scheme for mutual authentication instead of public key infrastructure (PKI). Another difference is that, the message exchanged in the mutual member key agreements do not allow the other members in the same subtree to generate the node key. The two members use unicast to exchange their public information and to establish the new node key.

To pass the new key on to the other members residing in the same subtree, both members encrypt the new key with the root key of their respective subtree, and broadcast it to the members in the same subtree, for example in Fig 2.16, if M_1 and M_3 perform the key agreement for the root key $K_{\langle 0,1\rangle}$, M_1 will use $K_{\langle 1,1\rangle}$ to encrypt the new key $K_{\langle 0,1\rangle}$ before sending it to the other members in the same subtree. In the same way, M_3 uses $K_{\langle 1,2\rangle}$ for encryption.

Join operation: In case of join event, all keys from the leaf node of a new member up to the root need to be changed. When a new member request for a join, the member nearest to the

root replied for this request (each member has the map of complete key tree). Therefore, the new member joins at the closest node to the root.

Leave operation: In case of leave operation, all the keys, that the leaving member knows, need to be changed, which prevents him or her from knowing any of the new keys. When the member at node $< l, i >$ leaves the group, the node $< l-1, \left\lceil \frac{i}{2} \right\rceil >$, which gets replaced by $< l, i \pm 1 >$ (by moving the entire subtree one level up). All keys from the deleted node up to the root to be updated: $K_{< l', \left\lceil \frac{i}{2^{l'}} \right\rceil >} (0 \le l' \le l)$.

The NAGKA protocol is an extension of two party Diffie-Hellman protocol, and does not support authentication. However, it is useful in certain environment where the group members prefer to remain anonymous. It may provide a good solution for mobile ad hoc networks.

2.3.8.4 The STR Protocol

The Skinny Tree (STR) protocol, proposed by steer et al. [28] and undertaken by Kim et al. in [114], is a contributive protocol using a tree structure. The leaves are associated to group members. Each leaf is identified by its position LN_i in the tree and holds a secret random R_i (generated by the corresponding member M_i) and its public blinded version $BR_i = \alpha^{R_i} \bmod p$, where α and p are DH parameters. Each internal node is identified by its position IN_i in the tree and holds a secret random K_i and its blinded public version $BK_i = \alpha^{K_i} \bmod p$. Each secret K_i is recursively calculated as follows:

$$K_i = (BK_{i-1})^{R_i} \bmod p = (BR_i)^{K_{i-1}} \bmod p$$

The group key $K_n = \alpha^{R_n \alpha^{R_{n-1} \ldots \alpha^{R_2 R_1}}}$ is the key associated to the root. Due to the linear structure of the tree, this solution induces a O(n) key calculations in order to establish the group key associated to the root of the tree. Besides, each member should store and maintain all the public keys associated to all the nodes of the tree. The STR protocol is an extension of two party Diffie-Hellman protocol [110]. This protocol is also based on tree approach and require few number of rounds, number of messages, and exponential operations, therefore, it may be suitable for mobile ad hoc networks.

2.3.9 Secret Share-based Cooperation

In this subcategory the nodes use secret share cooperation among themselves for creating the group key. The following subsections describe some protocols available in the literature based on this approach.

2.3.9.1 Bresson and Catalano's Group Key Agreement Protocol

A constant round provably authenticated static group key agreement protocol is introduced by Bresson and Catalano [30], which is based on secret sharing techniques combined with the ElGamal encryption scheme and uses asynchronous network. Their analysis is in the standard model under Decision Diffie-Hellman assumption.

This protocol is based secret sharing techniques and ElGamal encryption scheme and uses asynchronous network. Since we are exploring the group key agreement protocols based on two party Diffie-Hellman protocol [110], therefore, this protocol is not suitable further investigation.

2.4 Independent Group Key (GK) per Subgroup Approach

The common group key (GK) approach has the drawback to require that all group members commit to a new GK, whenever a membership change occurs in the group, in order to ensure perfect backward and forward secrecy. This is commonly called one-affects-all phenomenon. In order to mitigate the one-affects-all phenomenon, another approach consists in organizing group members into subgroups. Each subgroup uses its own independent group key (GK). In this scheme when a membership change occurs in a subgroup, it affects only the members belonging to that subgroup. The existing protocols that use independent group key (GK) per subgroup fall into two subcategories: the membership-driven re-keying protocols that do re-keying after each membership change, and time-driven re-keying protocols that do batch re-keying after each time interval. Fig 2.1 depicts this classification.

2.4.1 Membership-Driven Re-keying

In this subcategory of protocols, the existing group key is changed whenever a join or a leave operation occurs in the group. The main drawback of these protocols is that they perform a lot of computations when communication takes place among members of different

subgroups, which need computationally powerful nodes. The following subsections present some protocols available in the literature based on this approach.

2.4.1.1 Iolus

Mittra [96] proposed Iolus, a framework of a hierarchy of multicast subgroups. Each subgroup is an independent multicast group (with its own multicast address and eventually its own multicast routing protocol). The overall subgroups form a virtual multicast group. A Group Security Agent (GSA) manages each subgroup, which is responsible for key management inside the subgroup. A main controller called the Group Security Controller (GSC) manages the GSAs. Each of them uses its own group key, when a membership change occurs in a subgroup, only that subgroup is involved in a rekey process. This way, Iolus scales to large groups and mitigates one-affects-all phenomenon. However, Iolus has the drawback of affecting the data path. Indeed, there is a need for translating the data that goes from one subgroup, and thereby one key to another. This induces decryption / reencryption operations that are not tolerated by most of delay sensitive applications.

2.4.1.2 Keyed HIerarchical Multicast Protocol

Shields et al. [18] proposed the Keyed Hierarchical multicast Protocol (KHIP). KHIP is based on a multicast tree built using OCBT [19] routing protocol. It uses also an authentication service [58] based on certification to authenticate members and on-tree routers. The multicast tree is organized into sub-branches. Each sub-branch is managed by a trusted router, which manages the group key (GK) used within this sub-branch. When a source is ready to send a message to the group, it generates a random key RK, encrypts the message with RK , and encrypts RK with the GK of the sub-branch to which the source is attached. Then the source puts the encrypted RK in the header of the packet carrying the message and multicasts the packet. The members located in the same sub-branch know the GK of the sub-branch and hence can decrypt the RK and then decrypt the message with RK. When a border router of the sub-branch (a trusted router at the intersect between two sub-branches) receives the packet, it decrypts the RK and re-encrypts it using the GK of the adjacent sub-branch to which the so translated packet will be forwarded. This process is followed until the message reaches all the group members. When a new member joins a sub-group, the router responsible for that sub-branch generates and distributes a new GK for the sub-branch encrypted with the

old one. When a member leaves a sub-branch, the corresponding router distributes a new GK encrypted with the public key of each remaining member and signed with the router's private key. Even thought KHIP reduces the decryption / reencryption operations to a single key per packet, it still suffers from the delay variations of packet delivery due to these operations, and most of applications that require real-time transmission do not tolerate such delays.

2.4.1.3 Cipher Sequences (CS)

Molva and Pannetrat [85] proposed a framework for multicast security that is based on Reversible Cipher Sequences. A function $f(S, a)$ is called Cipher Group (CG) if it has the following characteristics: there is a sequence of n elements $a_i (1 \le i \le n)$, and there is a sequence of $n+1$ elements $S_i (1 \le i \le n)$, such as $S_i = f(S_{i-1}, a_i)$ for $i > 0$ and S_0 is the initial value, and for every pair (i, j) , where $i > j$, there exists a function $h_{i,j}$ such as $S_i = h_{i,j}(S_j)$. The multicast tree is rooted at the source, the group members are the leaves and internal nodes are intermediate elements of the multicast communication.

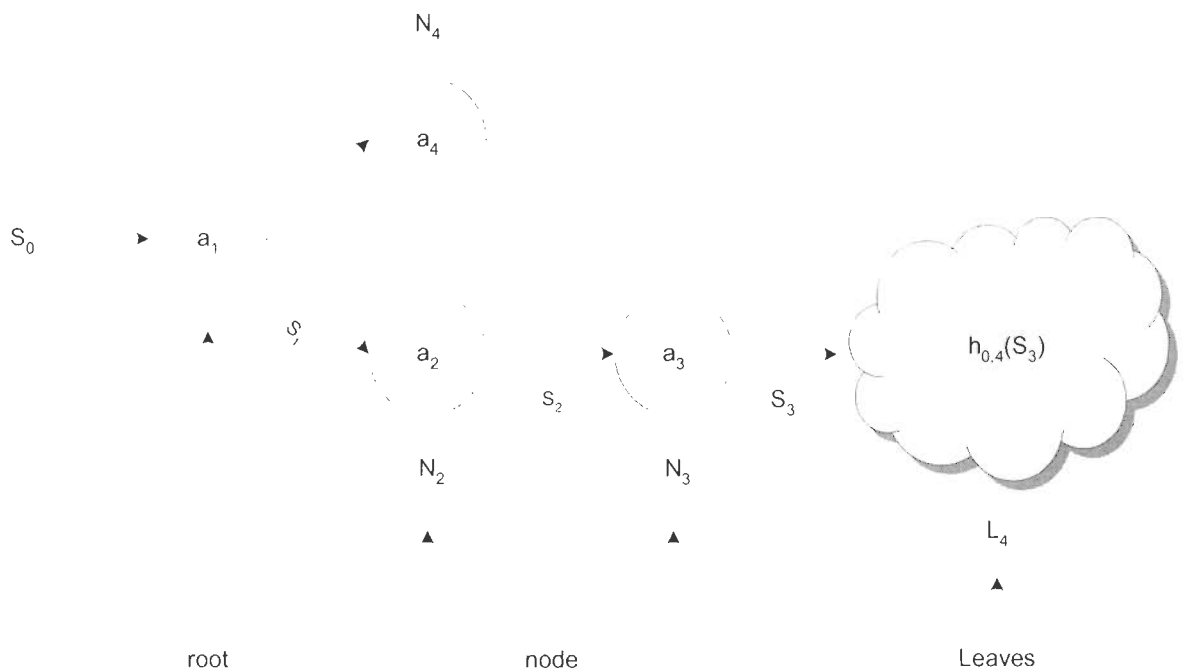


Figure 2.18: Cipher Sequence (CS) Protocol

Now, let S_0 be the message to be multicast and let every node N_i be assigned a value $a_i > 1$. When node N_i receives a value S_j from its parent N_j, it computes $S_i = f(S_j, a_i)$ and

sends S_i to its children that can be either leaves or other internal nodes. The leaves are assigned the function $h_{0,n}$, which enables them to compute S_0 from S_n, since $S_0 = h_{0,n}(S_n)$, and therefore recover the original data. Fig 2.18 typifies Molva scheme.

When a membership change occurs in a leaf, the corresponding node N_n receives a new value $a_n^{'}$ and all members in the leaf receives a new function $h_{0,n}^{'}$. Naturally, if the membership change occurred because of member removal the removed member would not receive the new $h_{0,n}^{'}$, thus will not be able to recover S_0.

2.4.1.4 Scalable and Adaptive Key Management (SAKM) Scheme

Challal et al. [112] proposed a new Scalable and Adaptive Key Management scheme (SAKM) that addresses the one-affects-all and re-keying overheads by taking into consideration the dynamic aspect of the group members. The organization of the group into clusters is updated periodically depending on the dynamism of the members during the secure session. Many studies [59] show that the membership behavior of group members in multicast sessions is likely to be not uniform through a large-scale group and during the whole session. Some parts of the network may be more dynamic than others during some periods of time and become more stable afterward. In such case, it would be interesting to use a protocol that restricts the re-key to the areas with frequent membership changes. Therefore, SAKM is very efficient in such situations. Simulation results show that SAKM scales well to large groups by minimizing the one-affects-all phenomenon, while it reduces the decryption / reencryption operations thanks to the adaptive dimensioning of the clusters depending on the membership dynamism.

2.4.2 Time-Driven Re-keying

In this subcategory of protocols, the group key is changed after each specific period of time. Thereby, the departing members are not excluded immediately from having access to the secure content. Similarly, new members are delayed up to the beginning of a new interval of time. This is a major drawback of these protocols when join and leave events happen frequently. Some protocols available in the literature based this approach are as follows:

2.4.2.1 Yang Protocol

Yang et al. [116] proposed a reliable re-keying approach. In the proposed architecture, the multicast group is organized into a set of subgroups and a Key Server (KS) manages each subgroup. The role of a KS is to re-key the members of its subgroup periodically. The overall KSs share a common KS secret key. When a KS receives a multicast message encrypted with its local group key (GK) (sent by one of its subgroup members), it decrypts it and re-encrypts it using the KS secret key. Then, it multicasts the re-encrypted message to the other KSs. In turn, these KSs decrypt the message using the KS secret key and re-encrypt it using their respective local GKs. Then, each KS multicasts the re-encrypted message to its subgroup.

2.4.2.2 SIM-KM Protocol

Mukherjee and Atwood [86,88,89] proposed a multicast key management infrastructure called SIM-KM. Scalable Infrastructure for Multicast Key Management. SIM-KM bases on subgrouping with message transformation by local controllers. In contrast to solutions based on subgrouping, SIM-KM uses proxy encryption [87] to transform data at the border of a subgroup. Proxy functions convert cipher text for one key into cipher text for another key without revealing secret decryption keys or clear text messages. This allows SIM-KM to do subgrouping with data transformation in order to limit the impact of re-keying, even though intermediaries are not trusted entities.

2.5 Conclusion

The existing group key establishment protocols can be categorized into three categories: centralized, decentralized and contributory. In centralized protocols only one central server is responsible for creation and distribution of keys. In decentralized protocols based on common group key approach, one central server creates and other additional servers help in distributing keys, this way other servers share the load of the central server. In another category of decentralized protocols, the group is divided into subgroups and each subgroup has its own subgroup key. Although this eliminates one-affect-all phenomena, yet it involves a lot of encryption/decryption cost. Since nodes in a MANET are not computationally powerful, no one node can work as central server for creating and distributing the keys. Therefore, these protocols are not suitable for secure and efficient group communication in mobile ad hoc networks. In contributory protocols each node contributes equally in group key formation, therefore, these protocols seem to be promising for mobile ad hoc networks. The

contributory protocols are also known as group key agreement protocols. Boyd and Nieto [13] proposed a constant round group key agreement protocol which does not provide forward secrecy that is why it is not suitable for dynamic environment. In [13], group members consist of one member called initiator and other members called responders. The initiator has a heavy burden caused by $(n-1)$ encryptions in a public cryptosystem and one signature generation. In a MANET, no node can take the role of initiator because of lack of computational power that is why this protocol is also not suitable for a MANET. Bresson and Catalano [30] have proposed group key agreement protocol based on standard secret sharing techniques with 2-round in the standard model. This protocol is inefficient from a point of view of the computation rate. Each member should perform more than $3n$ modular exponentiations, $3n$ modular multiplications, n signature generations and n signature verifications. For dynamic groups, Bresson et al. improved the protocol [33] into dynamic group key agreement protocol [31]. However, this protocol does not has constant round, each group member embeds its secret in the intermediate keying material and forwards the results generated with the secret to the next group member. This makes number of rounds in setup and join algorithms linear with respect to the number of group members. That is why it is not scalable. Bresson et al. [32] introduced a provably secure authenticated group key distribution protocol with two round in the random oracle model [67], which is suitable for restricted power devices and wireless environments. They have concentrated on an efficient computation rate of a group member with a mobile device. In this protocol, however, there exists a base station as a trustee which is prone to different kind of attacks and performs large computations which ultimately makes it unsuitable for a MANET. Hence, it is concluded from the detailed analysis of group key agreement protocols, the protocols based on two party Diffie-Hellman protocol [110] can only provide a suitable solution for mobile ad hoc networks. Hereafter, the work concentrates on the group key agreement protocols, which are based on two party Diffie-Hellman protocol [110].

Chapter 3

Group Key Agreement in MANET

3.1 Introduction

Mobile ad hoc networks [10,17,42,44,63,82] have attracted significant attentions recently due to their wide applications in different areas. These networks do not have fixed infrastructure, such as switching centers or base stations. Mobile nodes that are within the communication range of each other can communicate directly whereas the nodes that are far apart have to rely on intermediary nodes (routers) to relay messages. The mobility of a node in the mobile ad hoc networks can cause frequent changes in the network topology. These networks are useful in applications such as military operations, relief and rescue operation in case of natural disaster etc. These are also very attractive option for commercial uses. Many multicast and group-oriented network applications can easily be conducted in this network environment, for example, in a conference room or in battlefield, users can form an ad-hoc network instantly with their wireless devices, e.g. notebook computers, Personal Digital Assistants (PDAs), or even cell phones, without requiring any pre-installed cables or base stations. They can use this fast setup ad-hoc network for conducting a videoconference, sharing files or even playing interactive games. For conducting these applications, a group key is often needed for the mobile nodes. This group key is created by group key establishment protocols. Nature of ad-hoc networks sets certain additional requirements for the group key establishment protocols, for example, fewer number of rounds, number of messages, and number of exponential operations.

This chapter analyzes many group key agreement protocols based on two party key agreement protocol of W. Diffie and M. Hellman [110]. According to this protocol, if two nodes A and B want to establish a shared key via an un-secure channel, they first agree on two large numbers, n and α where n is a prime and $(n-1)/2$ is also a prime. n and α can

be public. Then A picks a large number x and keeps it secret, and B also picks a secret number y. A computes $\alpha^x \bmod n$ and sends this to B. Similarly, B sends $\alpha^y \bmod n$ to A. On receiving $\alpha^y \bmod n$ from B, A computes $(\alpha^y \bmod n)^x \bmod n$. Similarly, B computes $(\alpha^x \bmod n)^y \bmod n$ where $\alpha^x \bmod n$ is from A. After the key exchange, both A and B have $\alpha^{xy} \bmod n$ and this is the established shared key $(\alpha^x \bmod n)^y \bmod n = (\alpha^x \bmod n)^y \bmod n = \alpha^{xy} \bmod n$. The security of Diffie-Hellman key exchange lies in the fact that, deducing x from $\alpha^x \bmod n$ is hard. So even if an eavesdropper has $\alpha^x \bmod n$, $\alpha^y \bmod n, \alpha$ and n, he still cannot deduce either x or y, and thus cannot compute the shared key $\alpha^{xy} \bmod n$. This is also called discrete logarithm problem. Despite its elegance, Diffie-Hellman key exchange is vulnerable to bucket-brigade attack or man-in-the-middle attack..

The chapter describes various existing group key agreement protocols briefly, based on the two party Diffie-Hellman key agreement [110] and present the complexity analysis of these protocols to assess their suitability for an ad hoc network.

3.2. Group Key Agreement Protocols

In a contributory protocol all participants contribute equally in the key generation and guarantee for their part that the resulting key is fresh. Contributory key establishment is also called key agreement. Obviously, we also have to assume that the network is not split at the time of the group key agreement, i.e. that all nodes that contribute to the group key have a connection to each other. Contributory protocols are good when there are no previously agreed common secrets. The parties might not trust each other and need to be convinced that the generated key is fresh and random. The lack of third parties means that no one can be trusted to calculate a random key safely and to distribute it, even if such an entity can be appointed, it is a bottleneck in the protocol and it can be compromised or out of reach of other nodes. An obvious requirement for a secure group key agreement is that it is secure against both active and passive attacks. The pieces of keys exchanged during the protocol should not reveal information that leads to the compromise of the group key. Neither should it be possible for an outsider to take advantage of the protocol in order to input extraneous key material that would compromise the key or to attain compromising information by pretending to be a genuine protocol participant.

3.2.1. The INGM Protocol

The protocol shown in Fig 3.1 was proposed by Ingemarsson et al. in [46]. This protocol is one of the first natural extensions of two party Diffie-Hellman protocol [110]. It requires a synchronous startup and completes in $(n-1)$ rounds. The participants must be arranged in a logical ring. In a given round, every member raises the previously received intermediate key value to the power of its own exponent and forwards the result to the next member. After $(n-1)$ rounds everyone computes the same key K_n. The main disadvantages of this approach are the high number of messages, $n(n-1)$, exchanged and the high number of exponential operations, n^2, required.

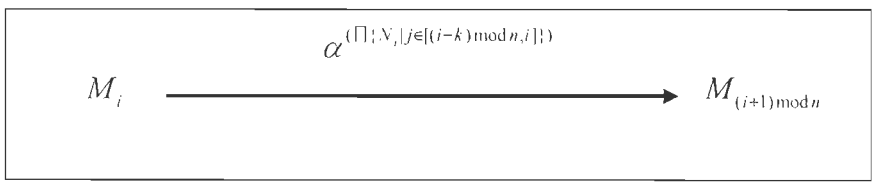


Figure 3.1. INGM Protocol: Round *k;* $k \in [1, n-1]$

3.2.2. The BD Protocol

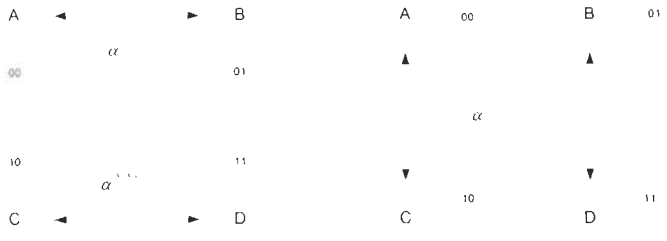
The BD protocol was presented by Burmester and Desmedt [71] and is executed in three rounds. Each member $M_i, i \in [1, n]$ performs the following operations:

- Each member M_i generates its random exponent N_i and broadcasts $z_i = \alpha^{N_i}$.
- Every member M_i computes and broadcasts $X_i = (z_{i+1} / z_{i-1})^{N_i}$.
- Each member M_i can now compute the key $K_n = z_{i-1}^{nN_i} . X_i^{n-1} . X_{i+1}^{n-2} ..... X_{i-2} \bmod p$

The group key has the form $K_n = \alpha^{N_1N_2+N_2N_3+.....+N_nN_1}$ and shares the security characteristics presented by the Diffie-Hellman algorithm [110]. This protocol is efficient with respect to the total number of rounds. However, it is costly in terms of number of exponential operations. This protocol requires $3, 2n(n-1)$ and $n(n-1)$ number of rounds, number of messages and number of exponential operations respectively.

3.2.3. The Hypercube and Octopus Protocols

Becker and Willie [12] proposed the Hypercube Protocol. It was designed to overcome the high number of messages needed by INGM protocol in [46] by logically arranging the nodes in a hypercube. The idea behind the hypercube key agreement approach is shown in Fig 3.2 with four members A, B, C, D who want to agree on a key. Each of them is given a two bits address 00, 01, 10, 11 respectively. Let contribution by each member to a two party Diffie-Hellman be N_A, N_B, N_C, and N_D.



(a) **(b)**

Figure 3.2 : Pairwise exchange in a d-cube (a) round 1 (b) round 2

In the first round as depicted by Fig 3.2(a) A and B engage a two party Diffie-Hellman protocol and calculate the key $N_{AB} = \alpha^{N_A N_B}$; C and D similarly calculate the key $N_{CD} = \alpha^{N_C N_D}$. In the next round as depicted by Fig 3.2(b) A and C participate in the Diffie-Hellman protocol using N_{AB} and N_{CD} as random numbers instead of selecting new random numbers. In other words, they will generate a key $\alpha^{N_{AB}N_{CD}}$. Similarly, B and D also participate the Diffie-Hellman protocol to get the key $\alpha^{N_{AB}N_{CD}}$. Thus, the final key calculated by all members is $N_{ABCD} = \alpha^{N_{AB}N_{CD}}$.

For simplicity, It is assumed that the number of members is $n = 2^d$. Each member is assigned a vertex and a unique d-bit address from the set Z_n. The protocol runs for d rounds. In the j^{th} round, neighbors along the j^{th} dimension of the hypercube participate in a two party Diffie-Hellman protocol. After d rounds all members share the same key. The hypercube protocol needs $\lceil \log_2 n \rceil, n\lceil \log_2 n \rceil$ and $2n\lceil \log_2 n \rceil$ number of rounds, number of messages, and number of exponential operations respectively for its completion. Becker and Wille [12] also discussed a protocol named octopus, in which members are divided into four disjoint subgroups, and each subgroup has a member as the header. Let A, B, C, D be such

four group headers. Member A first builds a secure communication channel with each of its subgroup members A_i. Then A_i generates a random k_i and sends it to A. The member, A, computes $N_A = \prod_{A_i \in G(A)} k_i$, where G($A$) is the subgroup headed by A. Header B, C and D also generate N_B, N_C, and N_D. Then A, B, C, D performs a hypercube key agreement protocol described above to get a common key $K = \alpha^{N_B N_C}$. Then A sends each of its subgroup members A_i the following values: $x_i = (\alpha^{N_B})^{N_A k_i}$ and $y = \alpha^{N_A N_C}$. The member A knows N_{AB}, k_i, α^{N_B}, and $\alpha^{N_A N_C}$. Then member A_i first calculates $N_{AB} = x_i^{k_i}$ and the final key as $y^{N_{AB}} = \alpha^{N_{AB} N_A N_C}$, which is same for all group members. The octopus protocol needs $(2\left\lceil \frac{n-1}{4} \right\rceil + 2), (3n-4)$, and $(\frac{n+12}{4})$ number of rounds, number of messages and number of exponential operations respectively for its completion.

3.2.4. The CLIQUES Protocol Suite

Steiner M. et al. proposed CLIQUES protocol suite in [77] based on [76], that consist of key management protocols for dynamic groups. Two of these protocols, IKA.1 and IKA.2 (Initial Key Agreement 1 and 2), are defined for group key establishment. Other protocols are specified for member and subgroup addition and exclusion and key refresh.

3.2.4.1. IKA.1. At the first stage, as shown in Fig 3.3, contributions are collected from all group members through $(n-1)$ rounds. Each group member (except the first) receives a data set that represents the partial contributions from all the group members that have already executed this first stage. The member adds its contribution and sends a new data set to the next group member. As an example, node M_4 receives the set $\{\alpha^{N_1 N_2 N_3}, \alpha^{N_1 N_2}, \alpha^{N_1 N_3}, \alpha^{N_2 N_3}\}$, and sends the set $\{\alpha^{N_1 N_2 N_3 N_4}, \alpha^{N_1 N_2 N_4}, \alpha^{N_1 N_2 N_3}, \alpha^{N_1 N_3 N_4}, \alpha^{N_2 N_3 N_4}\}$ to M_5. The set sent by the i^{th} node consists of i intermediate values, each containing $(i-1)$ exponents, and a cardinal value containing i exponents that correspond to the exponentiation base raised to every contribution generated so far. The last group member, M_n, is called the group controller. At the end of the first stage it receives a data set whose cardinal value is $\alpha^{N_1 N_2 \ldots N_{n-1}}$ and computes the group key $K_n = \alpha^{N_1 N_2 \ldots N_n}$.

At the second stage as shown in Fig 3.3 the group controller adds its contribution to each intermediate value and broadcasts this new data set to every other node in the network. Each intermediate value now consists of the contribution of all group members except one. In order to compute the group key, each group member M_i identifies the appropriate intermediate value (the one that does not contain its contribution) and raises it to its contribution N_i obtaining K_n. The IKA.1 protocol needs n, n and $(n/2(n+3)-1)$ number of rounds, number of messages and number of exponential operations respectively for its' completion.

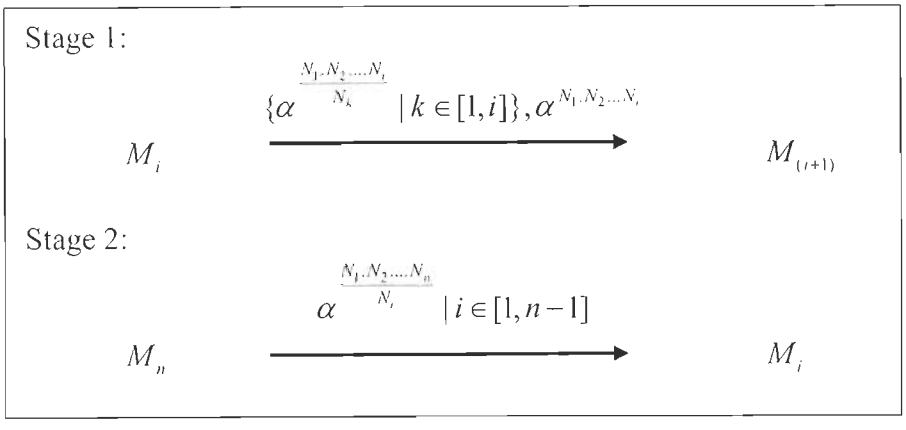


Figure 3.3. Stage 1 and 2 of IKA.1 Protocol

3.2.4.2. IKA.2. In IKA.1 protocol, the i^{th} node does $i+1$ exponential operations. However, in environments where nodes are having limited computational power, it is desirable to minimize the computational effort demanded from each group member. The IKA.2 protocol was proposed in order to minimize the demanded computational cost. It is similar to IKA.1 protocol but is executed in four stages as shown in Fig 3.4. In the first stage, as shown in Fig 3.4, contributions are collected from the $(n-2)$ first group members by means of a single message sent from one member to the next that gathers all the previous contributions. In the second stage, M_{n-1} adds its contribution to the received message and broadcasts this new message to the $(n-2)$ first members. In the third stage each member factors out its own contribution and sends this result to the last group member. In the last stage, M_n collects all the sets from the previous stage, raises each one of them to its contribution N_n and broadcasts these results to the other group members, allowing them to compute the group key.

These protocols have the advantage of requiring a low number of messages. The IKA.2 protocol has reduced the number of exponential operations required for group key establishment. Unlike the other presented protocols, this protocol suite provides mechanisms for group addition and exclusion, making it unnecessary to execute the entire key establishment protocol. This characteristic reduces the involved costs and provides backward and forward confidentiality. The IKA.2 protocol needs $(n+1),(2n-1)$ and $(5n-6)$ number of rounds, number of messages and number of exponential operations respectively for its' completion.

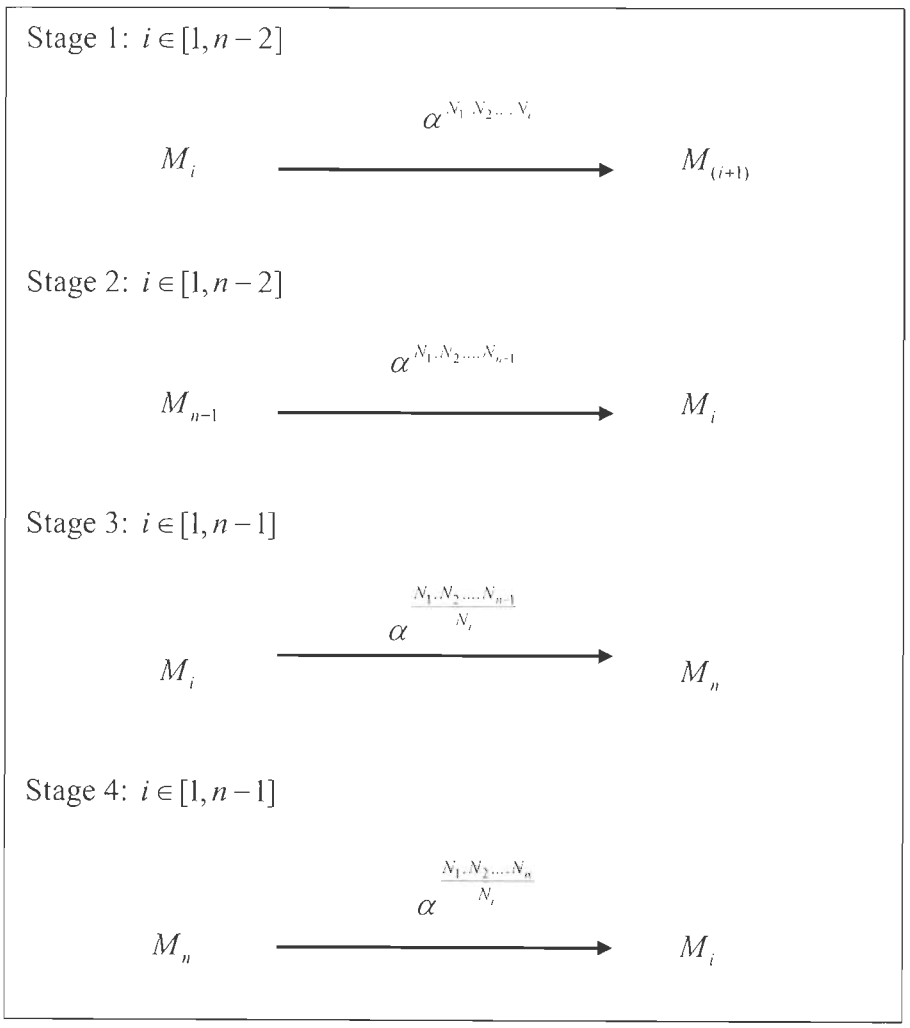


Figure 3.4. Stage 1, 2, 3 and 4 of IKA.2 Protocol

3.2.5. The TGDH Protocol

Wallner et al. first presented the binary tree-based approach for key establishment in [29]. Wong et al. then extended it for multiple degree tree in [15]. Yongdae Kim et al. proposed

Tree-Based Group Diffie-Hellman (TGDH) protocol in [113,115]. TGDH combines a binary tree structure with the group Diffie-Hellman technique as shown in Fig 3.5. They used one-way functions to enhance the security of protocol. Each node x in the key tree has two cryptographic keys, a secret key k_x and a blinded key $b_x = g(k_x)$, i.e., the blinded key b_x is computed from the secret key k_x using the one-way function g. The key is blinded in the sense that an adversary with limited computational capability can know b_x but cannot find k_x.

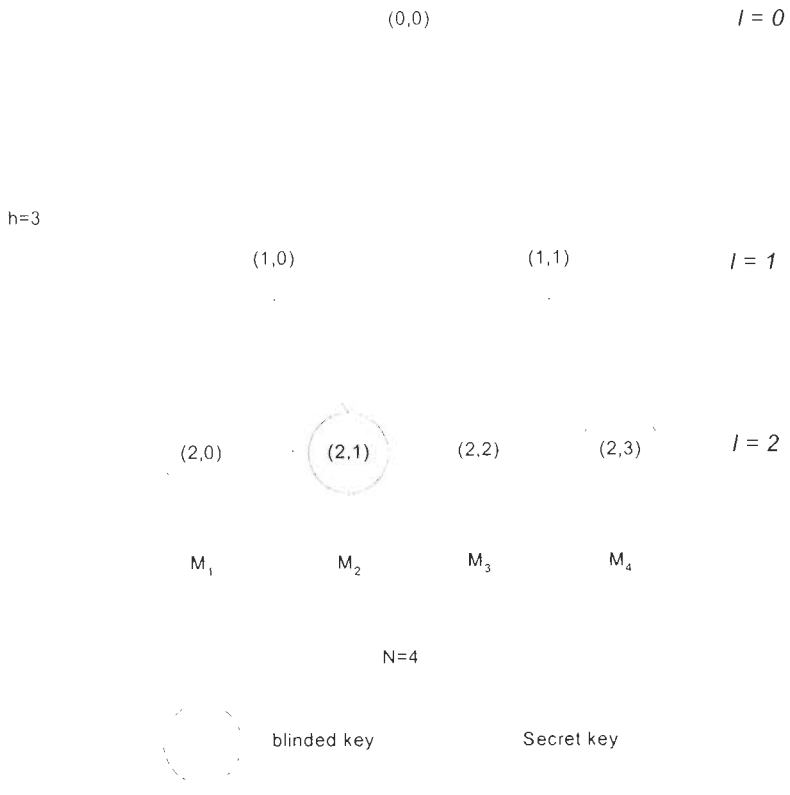


Figure 3.5. Tree-based Group Diffie-Hellman (TGDH) Protocol

The TGDH protocol uses the hierarchy in a binary tree to its advantage. The root is at the topmost level, given a value of 0 and all the leaves are at the lowest level h. Since the key tree is a binary tree, therefore, each node is either a leaf or a parent of two nodes. Each leaf node in the tree represents a group member M_i. The internal nodes are used for the key management and do not represent any individual member. Each node of the tree is represented by (l, v), where l is its level in the tree and v is the index of this node in level l. The key associated to node (l, v) is $k_{(l,v)}$ and its blinded key $b_{(l,v)} = \alpha^{k_{(l,v)}} \bmod p$. Each group

member contributes equally to the group key. In other words, for each internal node (l, v), its associated key $k_{(l,v)}$ is derived from its children's keys where $k_{(l,v)} = b^{k_{(l+1,2v)}}_{(l+1,2v+1)}$, which is essentially $\alpha^{k_{(l+1,2v+1)} \cdot k_{(l+1,2v)}}$. The group key is a result of contributions by the current members. The finally key at the root $(0,0)$ is computed and hashed, which results in group key. The TGDH protocol needs $(\lceil \log_2 n \rceil), (n\lceil \log_2 n \rceil / 2)$ and $(3n\lceil \log_2 n \rceil / 2)$ number of rounds, number of messages and number of exponential operations respectively for its' completion.

3.2.6. The NAGKA Protocol

Adrian Perrig [4] proposed a non-authenticated group key agreement (NAGKA) protocol. The lack of authentication can be beneficial in many settings, for example, when group members prefer to remain anonymous, or when the members do not share a commonly trusted third party.

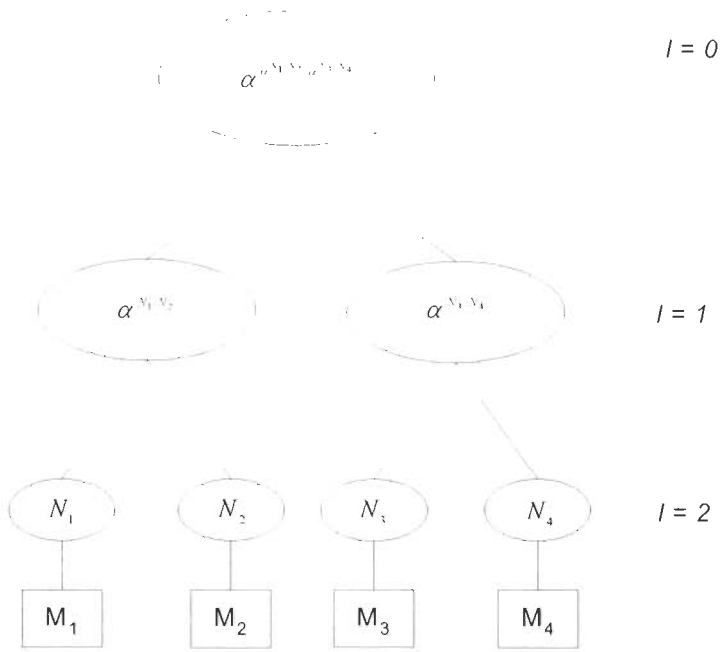


Figure 3.6. Key Tree (depth = 2) in NAGKA Protocol

This protocol is a variation of the non-authenticated two party Diffie-Hellman key agreement [110], extended to group agreement. According to the NAGKA protocol the key tree is constructed from the leaves up to the root as shown in Fig 3.6. Initially each member chooses a random number, for the key value of its leaf node, which collectively forms the lowest level of the key tree. For the construction of each subsequent layer, two members

establish a key through the key agreement procedure, one member from the left subtree and the other from the right subtree. They both broadcast their part of the Diffie-Hellman key agreement, which allows all the members of the subtree to compute the key of the current level. Once the root key is established, the group can start the secure communication by encrypting all messages with the root key. The NAGKA protocol needs $(\lceil \log_2 n \rceil), (2n-1)$ and $n(\log_2 n + 2)$ number of rounds, number of messages and number of exponential operations respectively for its' completion.

3.2.7. The STR Protocol

The Skinny Tree (STR) protocol, proposed by steer et al. [28] and undertaken by Kim et al. in [114], is a contributive protocol using a tree structure. Fig 3.7 depicts an STR tree with three members.

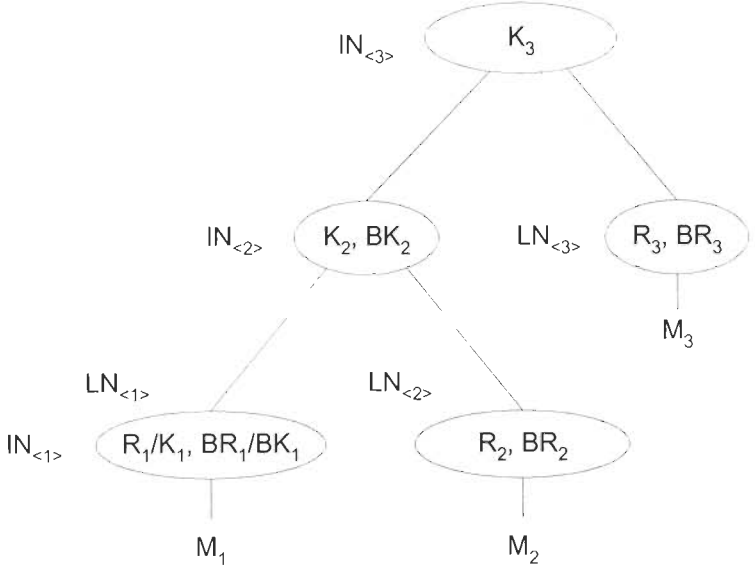


Figure 3.7: STR Key Tree with Three Members

The leaves are associated to group members. Each leaf is identified by its position LN_i in the tree and holds a secret random R_i (generated by the corresponding member M_i) and its public blinded version $BR_i = \alpha^{R_i} \bmod p$, where α and p are Diffie-Hellman parameters. Each internal node is identified by its position IN_i in the tree and holds a secret

random K_i and its blinded public version $BK_i = \alpha^{K_i} \bmod p$. Each secret K_i is recursively calculated as follows:

$$K_i = (BK_{i-1})^{R_i} \bmod p = (BR_i)^{K_{i-1}} \bmod p$$

The group key $K_n = \alpha^{R_n \alpha^{R_{n-1} \cdots \alpha^{R_2 K_1}}}$ is the key associated to the root. Due to the linear structure of the tree, this solution induces a O(n) key calculations in order to establish the group key associated to the root of the tree. Besides, each member should store and maintain all the public keys associated to all the nodes of the tree. The STR protocol needs $2n, 3n$ and $4n$ number of rounds, number of messages and number of exponential operations respectively for its' completion.

3.3. Complexity Analysis

Table 3.1 presents a comparative analysis of various group key agreement protocols based on complexity parameters such as number of rounds, number of messages communicated, number of exponential operations performed and round synchronization. Here n denotes the number of members in the group.

The INGM and BD protocols are most costly in terms of the number of messages and exponential operations, which make them very unattractive to use when compared to others for an ad hoc network. Moreover, INGM and BD protocols are not suitable for dynamic membership because these need to execute afresh whenever a dynamic event takes place. These protocols also need round synchronization. The round synchronization means a protocol round can only be initiated after the previous round has been completed. This leads to the need of a synchronizing mechanism that guarantees that all the participating devices received the messages intended to them at every round. The use of this kind of mechanism increases the execution time and the number of transmitted messages that varies with respect to the complexity of the synchronizing mechanism. Moreover, due to synchronizing mechanism some broadcast messages emanate simultaneously, which are problematic in mobile ad hoc networks. TGDH protocol is not suitable for mobile ad hoc networks because of round synchronization, which involves a lot of modular exponentiations, which are most expensive operation. It also requires extra cost for hash computation, and each node has to store the key tree, which again requires a lot of memory.

Table 3.1. Comparison of Group Key Agreement Protocols

Key Agreement Protocol	Number of Rounds	Number of Massages	Exponential Operations	Round Synchronization
INGM	$n-1$	$n(n-1)$	n^2	Yes
BD	3	$2n(n-1)$	$n(n-1)$	Yes
Hypercube	$\lceil \log_2 n \rceil$	$n.\lceil \log_2 n \rceil$	$2n.\lceil \log_2 n \rceil$	Yes
Octopus	$2.\left\lceil \frac{n-1}{4} \right\rceil + 2$	$(3n-4)$	$(n+12)/4$	Yes
IKA.1	n	n	$(n/2)(n+3)-1$	No
IKA.2	$n+1$	$2n-1$	$5n-6$	No
TGDH	$\lceil \log_2 n \rceil$	$n.\lceil \log_2 n \rceil/2$	$3n.\lceil \log_2 n \rceil/2$	Yes
NAGKA	$\lceil \log_2 n \rceil$	$2(n-1)$	$n(\log_2 n+2)$	Yes
STR	$2n$	$3n$	$4n$	Yes

NAGKA protocol is also tree-based, but it needs subgroup leader in each round, which is vulnerable to different kind of attacks and it also involve round synchronization. STR protocol looks good, but it also needs round synchronization and also involves parallel broadcasts of blinded version of secret keys due to synchronizing mechanism among nodes. The hypercube and octopus protocols are good in first three parameters, but these protocols again demand round synchronization similar to other protocols except Clique suite. The protocols in CLIQUES Suite present some advantages in all these areas. The IKA.2 protocol has the best overall performance regarding the number of rounds, messages exchanged, and number of exponential operations performed for establishing a group key among nodes. Although the IKA.1 protocol is not as efficient regarding these parameters, its use as an alternative to the IKA.2 protocol may be interesting since it is more efficient in terms of the number of messages. Both protocols belong to the same protocol suite, and complexity analysis shows that the final group key can be determined by any of these protocols.

3.4. Conclusion

The chapter examined the various group key agreement protocols, based on two party Diffie-Hellman protocol [110], for their suitability for mobile ad hoc network.

The protocols from the CLIQUES protocol suite, which do not need round synchronization, are among the ones' with best performance for MANET. And are the only ones to provide good performance on different parameters. Due to these reasons, CLIQUES protocol suite is the most appropriate for group key agreement in mobile ad hoc networks. This protocol suite requires sequencing among the group members that defines in particular the sequence of group members the contributions must go through and the last node in the sequence that acts as the group controller. The way by which this sequencing must be established is not defined and can be left to the designer. This sequencing can be fixed, using a predefined sequence, or be determined during the execution of the group key agreement protocol.

Chapter 4

Dynamic membership

4.1 Introduction

MANETs have many peculiarities such as absence of a fixed network infrastructure, frequent and unpredictable disconnections, bandwidth limitations, and power limitations. These render the development of ad hoc mobile applications a very challenging task. Since the radio links are very fragile in ad hoc networks, these often cause partitions in the networks. And, possibly after healing of the links in the network, groups merge together. Therefore, the dynamic group key agreement protocols for mobile ad networks pose more stringent requirements for dynamic events such as join, leave, merge and partition. Since members join (merge) or leave (partition) at their will or may be forced to do so due to network partitioning, the dynamic group key agreement protocols must be efficient in different parameters such as number of rounds, number of messages exchanged, and exponential operations.

The objective of this chapter is to describe briefly various existing dynamic group key agreement protocols based on the Two Party Diffie-Hellman Key Agreement Protocol [110], and analyze them for efficient dynamic membership events to assess the suitability for mobile ad hoc networks.

4.2. Secure Group Key Agreement Protocols

The general goal of secure group communication is to establish a common secret key (also referred to as a group key), among all group members for confidential communication [59]. Generally, a secret group key is established by a group key agreement protocol. Joining and leaving members pose the problem of backward and forward secrecy [15]. A protocol provides backward secrecy if a member joining the group at time t does not gain any

information about the content of messages communicated at times $t' < t$. A protocol provides forward secrecy if a member leaving the group at time t does not gain any information about the content of messages communicated at times $t' > t$. Protocols need to satisfy these properties to provide secure group communication in dynamic groups (dynamic implies that the membership can change through join, leave, merge, and partition events).

4.3. The CLIQUES Protocol Suite

Steiner M. et al. proposed CLIQUES protocol suite in [77] consisting of group key management protocols for dynamic groups. Two of these protocols, IKA.1 and IKA.2 (Initial Key Agreement 1 and 2), are defined for group key establishment.

4.3.1. IKA.1.

This protocol has two stages. In the first stage, contributions are collected from all group members through $n-1$ rounds. Each group member (except the first) receives a data set that represents the partial contributions from all the group members that have already executed this first stage. The member adds its contribution and sends a new data set to the next group member. The set sent by the i^{th} node consists of i intermediate values, each containing $i-1$ exponents, and a cardinal value containing i exponents that correspond to the exponentiation base raised to every contribution generated so far. The last group member, M_n, is called the group controller. At the end of the first stage it receives a data set whose cardinal value is $\alpha^{N_1 \cdot N_2 \cdots N_{n-1}}$ and computes the group key $K_n = \alpha^{N_1 \cdot N_2 \cdots N_n}$.

In the second stage, the group controller adds its contribution to each intermediate value and broadcasts this new data set to every other node in the network. Each intermediate value now consists of the contribution of all group members except one. In order to compute the group key, each group member M_i identifies the appropriate intermediate value (the one that does not contain its contribution) and raises it to its contribution N_i obtaining K_n.

4.3.1.1. Single join event.

For single join event, it is assumed that last member M_n saves the contents of the up-flow message in stage 1 (round $n-1$) during setup phase. When a new member joins, M_n generates

a new contribution $N_n^{'}$ and computes a new up-flow message including $N_n^{'}$ in place of N_n, $\{\alpha^{\frac{N_1.N_2...N_n^{'}}{N_k}} \mid k \in [1,n]\}, \alpha^{N_1.N_2....N_n^{'}}$ and sends it to the new member M_{n+1}, which in turn performs the same sequence of steps as M_n in IKA.1 protocol. The single join event requires $2,2$ and $(n+1)$ number of rounds, number of messages, and number of exponential operations respectively, where n denotes the number of existing members in the group.

4.3.1.2. Group fusion event.

This protocol assumes group fusion event as an event of multiple joins. When a group fusion event is identified, the current group controller generates new message including its new contribution and sends it to the one of the new members. This new member adds its contribution and sends this message to next new member. Ultimately, message reaches to last new member, which broadcast this message without including its contribution. Upon receiving the broadcast message, each member in the chain can compute the group key by factoring out the relevant component and adding its own contribution. The group fusion event requires $(j+1),(j+1)$ and $(\frac{1}{2}(j^2+2nj+j))$ number of rounds, number of messages, and number of exponential operations respectively, where j denotes the number of joining members.

4.3.1.3. Single leave event.

If member M_l is the leaving member, where $l \in [1,n-1]$, i.e., $l \neq n$. The last member M_n generates a new contribution $N_n^{'}$ and computes a new set of $n-2$ exponents excluding exponent of lest member as $\{\alpha^{\frac{N_1.N_2....N_n^{'}}{N_i}} \mid i \in [1,n-1] \wedge i \neq p\}$ and broadcasts them to all group members. Thus all remaining members can compute the group key. In case if current group controller M_n itself is leaving, then, member M_{n-1} assumes its role, which has required information (last Up-flow message to M_{n-1}) to perform a fast key update and relieve M_n. The single leave event requires $1,1$ and $(2n-1)$ number of rounds, number of messages, and number of exponential operations respectively where n denotes the number of existing members in the group.

4.3.1.4. Subgroup exclusion event.

This protocol is similar to single leave event protocol. In this case, when some members leave the group, the group controller removes their contributions from the message already stored with the group controller and broadcasts it to the remaining members in the group. Thus, remaining members can easily compute the group key. The notation n and l denote the number of existing members in the group and number of leaving members respectively. The subgroup exclusion event requires 1,1 and $(2n-l)$ number of rounds, number of messages, and number of exponential operations respectively.

4.3.2. IKA.2.

The IKA.2 protocol was proposed to minimize the computational load on nodes. It is similar to IKA.1 protocol but is executed in four stages. In the first stage, contributions are collected from the $n-2$ first group members by means of a single message sent from one member to the next that gathers all the previous contributions. In the second stage, M_{n-1} adds its contribution to the received message and broadcasts this new message to the $n-2$ first members. In the third stage each member factors out its' own contribution and sends this result to the last group member. In the last stage, M_n collects all the sets from the previous stage, raises each one of them to its contribution N_n and broadcasts these results to the other group members, allowing them to compute the group key.

4.3.2.1. Single join event.

It very similar to single join event in IKA.1 protocol. It is assumed that last member M_n saves the contents of the original broadcast and response messages (stage 2 & 3 in IKA.2 protocol). M_n generates a new contribution N_n' and computes a new up-flow message including N_n' in place of N_n , $\{\alpha^{\frac{N_1.N_2....N_n'}{N_k}} \mid k \in [1,n]\}, \alpha^{N_1.N_2...N_n'}$ and sends it to the new member M_{n+1}. It computes the new group key K_{n+1}. And after adding its contribution in each exponent of the received message, it broadcasts the message as in stage 4 of IKA.2, to all members. Now, all members can compute the shared group key. The single join event requires $4,(n+3)$ and $(n+3)$ number of rounds, number of messages, and number of

exponential operations respectively where n denotes the number of existing members in the group.

4.3.2.2. Group fusion event.

This protocol works similar up to the point when the members receive the broadcast from the last new member who ultimately becomes new group controller. Upon receiving the broadcast message, each member factors out its exponent and unicasts the result to the new group controller. The new group controller collects all the exponents and adds its contribution to each of them. It then broadcast all these exponents as single message to group members. Now every member computes the group key by factoring in its contribution. The notation n and j denote the number of existing members in the group and number of joining members respectively. The group fusion event requires $(j+3),(n+2j+1)$ and $(n+2j+1)$ number of rounds, number of messages, and number of exponential operations respectively.

4.3.2.3. Single leave event.

If member M_l is the leaving member where $l \in [1, n-1]$, i.e., $l \neq n$. The member M_n who saved the original broadcast and response messages (stage 2 & 3 in IKA.2 protocol) generates a new contribution $N_n^{'}$ and computes a new set of $n-2$ exponents excluding leaving member exponent, $\{\alpha^{\frac{N_1 \cdot N_2 \ldots N_n^{'}}{N_i}} \mid i \in [1, n-1] \wedge i \neq p\}$ and broadcasts them to remaining group members. Thus, all remaining members compute the group key. The single leave event requires 1,1 and $(n-1)$ number of rounds, number of messages, and number of exponential operations respectively where n denotes the number of existing members in the group.

4.3.2.4. Subgroup exclusion event.

This protocol is similar to single leave event protocol. In this case, when some members leave the group, the group controller removes their contributions from the message already stored with the group controller and includes a fresh contribution $N_n^{'}$ in place of N_n and broadcasts it to the remaining members in the group. Thus, remaining members can easily compute the group key. The notation n and l denote the number of existing members in the group and number of leaving members respectively. The subgroup exclusion event requires

1,1 and $(n-1)$ number of rounds, number of messages, and number of exponential operations respectively.

4.4. The TGDH Protocol

Yongdae Kim et al. proposed Tree-Based Group Diffie-Hellman (TGDH) protocol in [113,115]. In this protocol, each member maintains a set of keys, which are arranged in a hierarchical binary tree.

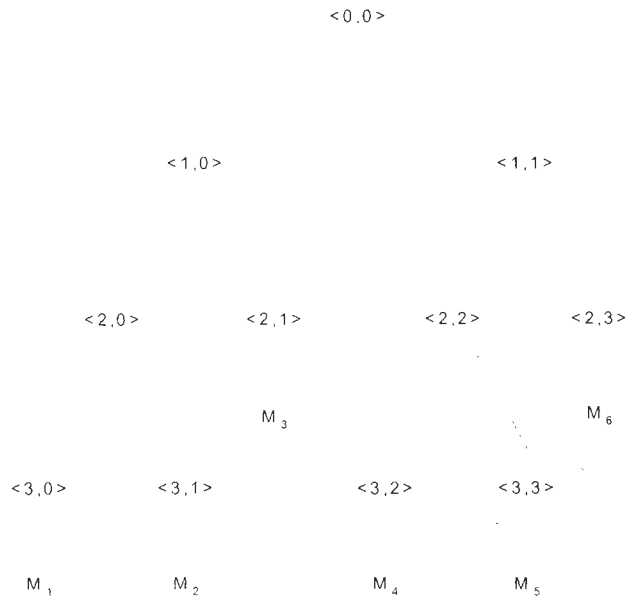


Figure 4.1: TGDH Key tree with six members

A node ID $<l,v>$ is assigned to every tree node. For a given node $<l,v>$, a secret (or private) key $K_{<l,v>}$ and a blinded (or public) key $BK_{<l,v>}$ is allocated. All arithmetic operations are performed in a cyclic group of prime order p with the generator α. Therefore, the blinded key of node v can be generated as $BK_{<l,v>} = \alpha^{K_{<l,v>}} \bmod p$

Each leaf node in the tree represents the individual secret and blinded keys of a group member M_i. Every member holds all secret keys along its key path starting from its associated leaf node up to the root node. Therefore, the secret key held by the root node is shared by all the members and is regarded as the group key. Fig 4.1 illustrates a possible key tree with six members M_1 to M_6. For example, member M_1 holds the keys at nodes $<3,0>$,

$<2,0>, <1,0>$, and $<0,0>$. And, at the same time, it also has all blinded keys on the key tree. The secret key at node $<0,0>$ is the group key of this peer group.

The node ID of the root node is set to $<0,0>$. Each non-leaf node consists of two child nodes whose node Ids are given by $<l+1,0>$ and $<l+1,1>$. Based on the two party Diffie-Hellman protocol [110], the secret key of a non-leaf node $<l,v>$ is generated by the secret key of one child node of non-leaf node and the blinded key of another child node of non-leaf node as $K_{<l,v>} = (BK_{<l+1,0>})^{K_{<l+1,1>}} \bmod p = (BK_{<l+1,1>})^{K_{<l+1,0>}} \bmod p = \alpha^{K_{<l+1,0>} K_{<l+1,1>}} \bmod p$.

Unlike the keys at non-leaf nodes, secret key at leaf node is selected by its' member through a secure pseudo random number generator. Since the blinded keys are publicly known, every member can compute the keys along its key path to the root node based on its individual secret key. To illustrate, consider the key tree in Fig 4.1. Every member M_i generates it's own secret key and all the secret keys along the path to the root node. For example, member M_1 generates the secret key $K_{<3,0>}$ and it can request the blinded key $BK_{<3,1>}$ from M_2, $BK_{<2,1>}$ from M_3, and $BK_{<1,1>}$ from either M_4, M_5 or M_6. Given M_1's secret key $K_{<3,0>}$ and the blinded key $BK_{<3,1>}$, M_1 can generate the secret key $K_{<2,0>}$ accordingly. Given the blinded key $BK_{<2,1>}$ and the newly generated secret key $K_{<2,0>}$, M_1 can generate the secret key $K_{<1,0>}$ accordingly. Given the secret key $K_{<1,0>}$ and the blinded key $BK_{<1,1>}$, M_1 can generate the secret key $K_{<0,0>}$ at the root. From that point onward, any communication in the group can be encrypted based on the secret key (or group key) $K_{<0,0>}$.

Let us first consider individual rekeying, meaning that rekeying is performed after every single join or leave event. Before the group membership is changed, a special member called the sponsor is elected to be responsible for updating the keys held by the new member (in the leave case). The convention used is that the rightmost member under the subtree rooted at the sibling of the join/leave nodes will take the sponsor role. The existence of a sponsor does not violate the decentralized requirement of the group key generation since the sponsor does not add extra contribution to the group key.

4.4.1. Single join event.

The Skinny Tree (STR) protocol, proposed by steer et al. [28] and undertaken by Kim et al. in [114], is a contributive protocol using a tree structure. Fig 4.6 depicts an STR tree with three members.

The leaves are associated to group members. Each leaf is identified by its position LN_i in the tree and holds a secret random R_i (generated by the corresponding member M_i) and its public blinded version $BR_i = \alpha^{R_i} \bmod p$, where α and p are DH parameters. Each internal node is identified by its position IN_i in the tree and holds a secret random K_i and its blinded public version $BK_i = \alpha^{K_i} \bmod p$. Each secret K_i is recursively calculated as follows:

$$K_i = (BK_{i-1})^{R_i} \bmod p = (BR_i)^{K_{i-1}} \bmod p$$

The group key $K_n = \alpha^{R_n \alpha^{R_{n-1}} + \ldots + \alpha^{R_2 R_1}}$ is the key associated to the root. Due to the linear structure of the tree, this solution induces a O(n) key calculations in order to establish the group key associated to the root of the tree. Besides, each member should store and maintain all the public keys associated to all the nodes of the tree. In case of a membership change (join/leave) the tree is re-built consequently and hence all the members update the group key, which is the new key K_n associated to the root of the tree.

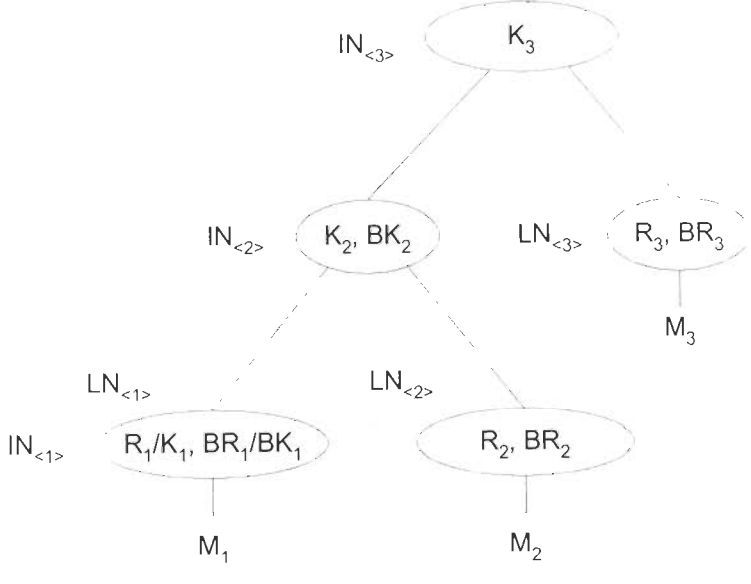


Figure 4.6: STR Key Tree with Three Members

4.5.1. Single join event.

It is assumed that the group has n users, $\{M_1.....M_n\}$ at the time of joining of new member. The new member M_{n+1} broadcasts a join request message that contains its own BK_{n+1} (Which is the same as its blinded session random BR_{n+1}).

Upon receiving this message, the joining group's sponsor M_n (that is, the last member already joined the group or top most member node in the current group) refreshes its session random, computes BR_n, K_n, BK_n, and sends the current tree BT_n to M_{n+1} with all blinded keys. Thereafter, each member M_i increments $n = n+1$ and creates a new root key node IN_n with two children, one root node IN_{n-1} of the prior tree T_i on the left and second the new leaf node LN_n corresponding to new member on the right. Now, every member can compute the group key since they all have new member's blinded session random. And, new joining member has the blinded group key of the joining group. In case of single join event, the sponsor is always the topmost leaf node, i.e., the most recent member joined the current group. Fig 4.7 depicts the joining of new member M_4. Here, M_3 is elected as sponsor that updates its session random R_3 for providing the forward and backward secrecy.

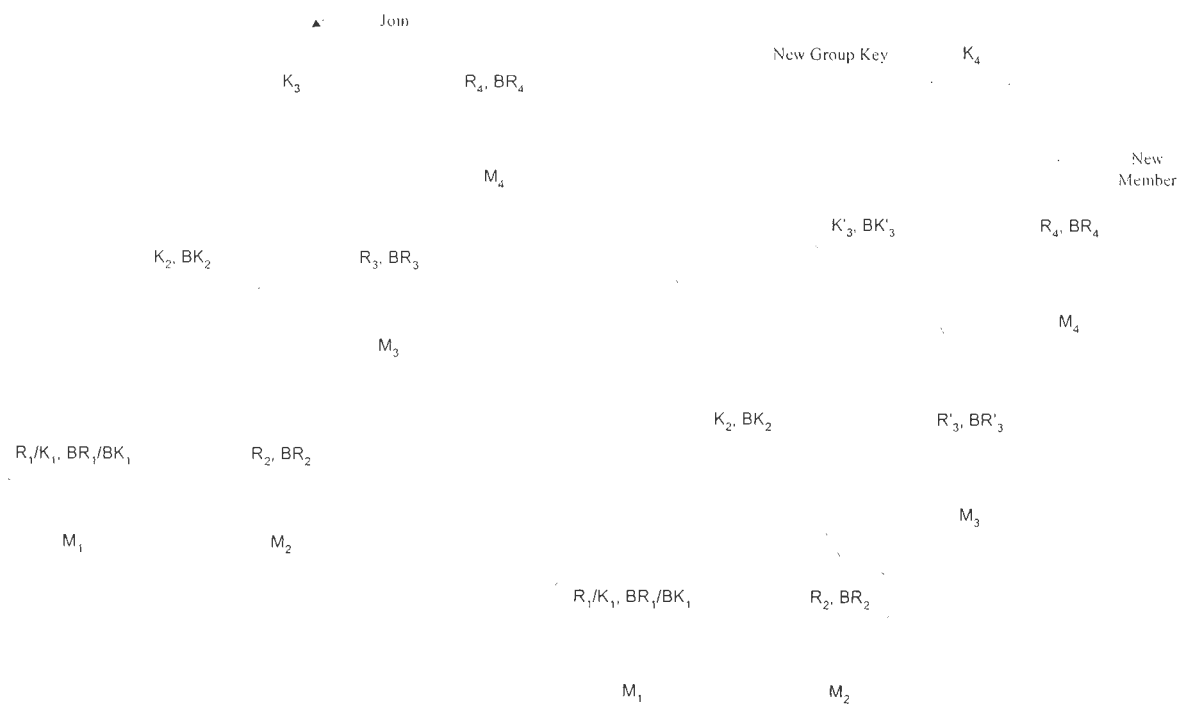


Figure 4.7: Single Join Event in STR Protocol

The single join event requires 1, 2 and 2 number of rounds, number of messages, and number of exponential operations respectively.

4.5.2. Merge event.

It is assumed that, as in the join event case, the group communication system simultaneously notifies all group members (in all groups) about the merge event. It is also assumed that smaller tree grafted atop the larger tree. And, if any two trees are of the same height, then, these can be merged based on some predefined criteria. When merging two trees, the lowest-numbered leaf of the smaller tree becomes the right child of a new intermediate node. The left child of the new intermediate node becomes the root of the larger tree. Using this technique recursively, multiple trees can be merged. In the first round of the merge protocol, all sponsors (members associated with the topmost leaf node in each tree) exchange their respective key trees containing all blinded session randoms. The highest-numbered member of the largest tree becomes the sponsor of the second round in the merge protocol. After refreshing its session random, this sponsor computes every secret key and blinded key pair up to the intermediate node just below the root node using the blinded session randoms of the other group members. It then broadcasts the key tree with the blinded keys and blinded session randoms to the other members. All members now have the complete set of blinded keys, which allows them to compute the new group key.

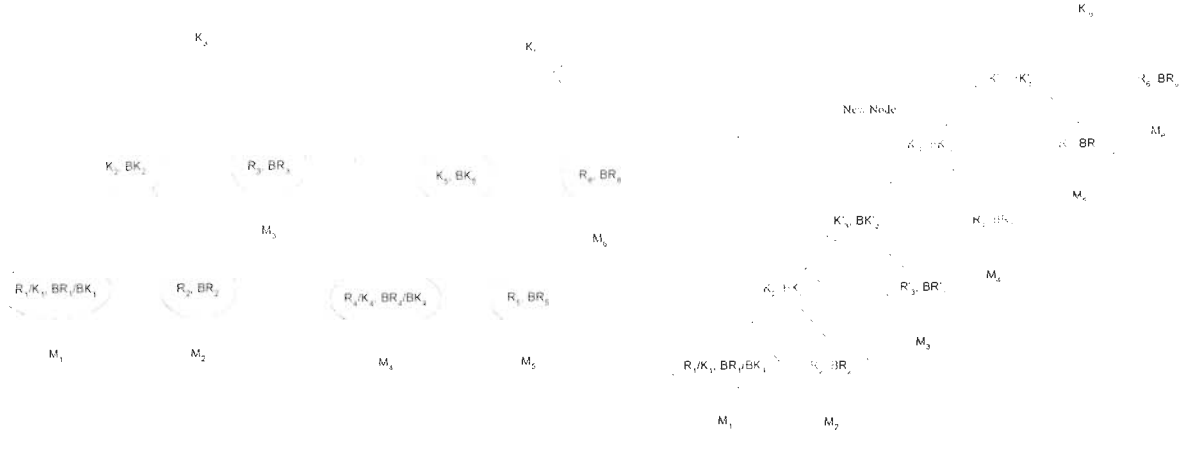


Figure 4.8: Merge Event in STR Protocol

$<2,0>$, $<1,0>$, and $<0,0>$. And, at the same time, it also has all blinded keys on the key tree. The secret key at node $<0,0>$ is the group key of this peer group.

The node ID of the root node is set to $<0,0>$. Each non-leaf node consists of two child nodes whose node Ids are given by $<l+1,0>$ and $<l+1,1>$. Based on the two party Diffie-Hellman protocol [110], the secret key of a non-leaf node $<l,v>$ is generated by the secret key of one child node of non-leaf node and the blinded key of another child node of non-leaf node as $K_{<l,v>} = (BK_{<l+1,0>})^{K_{<l+1,1>}} \bmod p = (BK_{<l+1,1>})^{K_{<l+1,0>}} \bmod p = \alpha^{K_{<l+1,0>} K_{<l+1,1>}} \bmod p$.

Unlike the keys at non-leaf nodes, secret key at leaf node is selected by its' member through a secure pseudo random number generator. Since the blinded keys are publicly known, every member can compute the keys along its key path to the root node based on its individual secret key. To illustrate, consider the key tree in Fig 4.1. Every member M_i generates it's own secret key and all the secret keys along the path to the root node. For example, member M_1 generates the secret key $K_{<3,0>}$ and it can request the blinded key $BK_{<3,1>}$ from M_2, $BK_{<2,1>}$ from M_3, and $BK_{<1,1>}$ from either M_4, M_5 or M_6. Given M_1's secret key $K_{<3,0>}$ and the blinded key $BK_{<3,1>}$, M_1 can generate the secret key $K_{<2,0>}$ accordingly. Given the blinded key $BK_{<2,1>}$ and the newly generated secret key $K_{<2,0>}$, M_1 can generate the secret key $K_{<1,0>}$ accordingly. Given the secret key $K_{<1,0>}$ and the blinded key $BK_{<1,1>}$, M_1 can generate the secret key $K_{<0,0>}$ at the root. From that point onward, any communication in the group can be encrypted based on the secret key (or group key) $K_{<0,0>}$.

Let us first consider individual rekeying, meaning that rekeying is performed after every single join or leave event. Before the group membership is changed, a special member called the sponsor is elected to be responsible for updating the keys held by the new member (in the leave case). The convention used is that the rightmost member under the subtree rooted at the sibling of the join/leave nodes will take the sponsor role. The existence of a sponsor does not violate the decentralized requirement of the group key generation since the sponsor does not add extra contribution to the group key.

4.4.1. Single join event.

Single member join event is illustrated in Fig 4.2. New member M_8 is the prospective member who wishes to join the group.

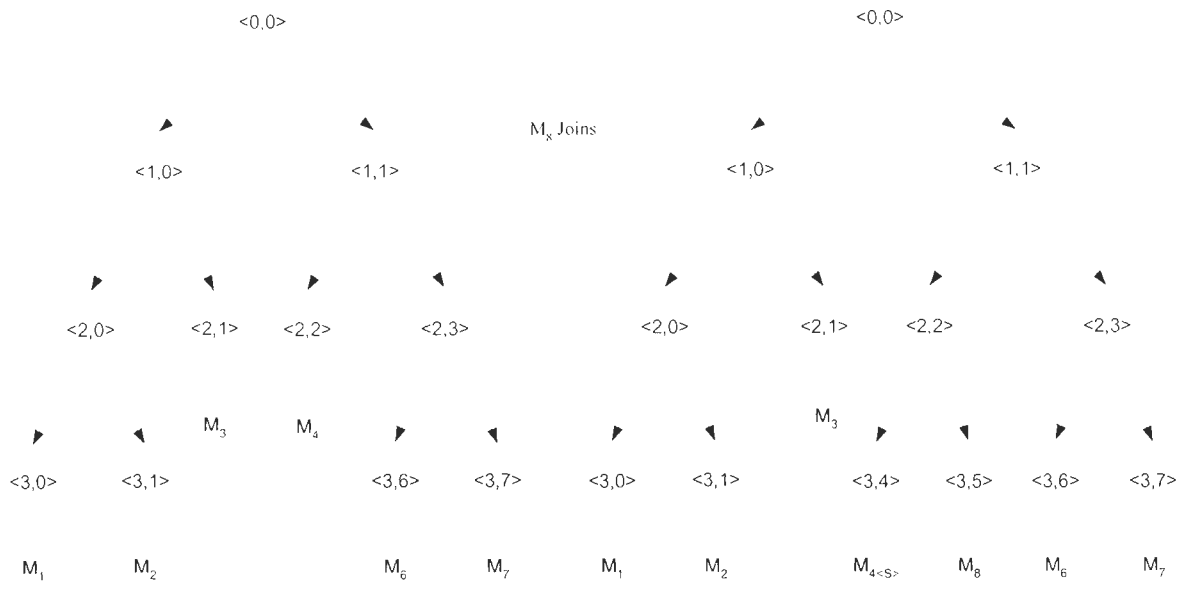


Figure 4.2: Single Join Event in TGDH Protocol

Initially, M_8 determines the insertion node under which M_8 can be inserted. To add a node, say v (or tree, say T) to the insertion node, a new node, say n, is first created. Then the subtree rooted at the insertion node becomes the left child of the node n, and the node v (or the root node of the tree T) becomes the right child of the node n. The node n will replace the original location of the insertion node. The insertion node is either the rightmost shallowest position such that the join does not increase the tree height, or the root node if the tree is initially well balanced (in this case, the height of the resulting tree will be increased by 1). Fig 4.2 illustrates this concept. The insertion node is node $< 2,2 >$ and the sponsor is M_4. Then, M_8 broadcasts its blinded key $BK_{<3,5>}$ upon insertion. Given $BK_{<3,5>}$, M_4 renews $K_{<2,2>}$ $K_{<1,1>}$ and $K_{<0,0>}$, and then broadcasts the blinded keys $BK_{<2,2>}$ and $BK_{<1,1>}$ to all members in the group. After receiving the blinded keys from M_4, all remaining members can rekey all the nodes along their key paths and compute the new group key $K_{<0,0>}$. The single join event requires 2,3 and $2\lceil \log_2 n \rceil$ number of rounds, number of messages, and number

of exponential operations respectively where n denotes the number of existing members in the group.

4.4.2. Merge event.

In case of merge event, the protocol works as follows: each sponsor (the rightmost member of each group) broadcasts its tree information to the merging subgroup after refreshing its session random and blinded keys. Upon receiving this message, all members uniquely and independently determine the merge position of the two trees (choose the joining node as the rightmost shallowest node, which does not increase the height of the resultant key tree).

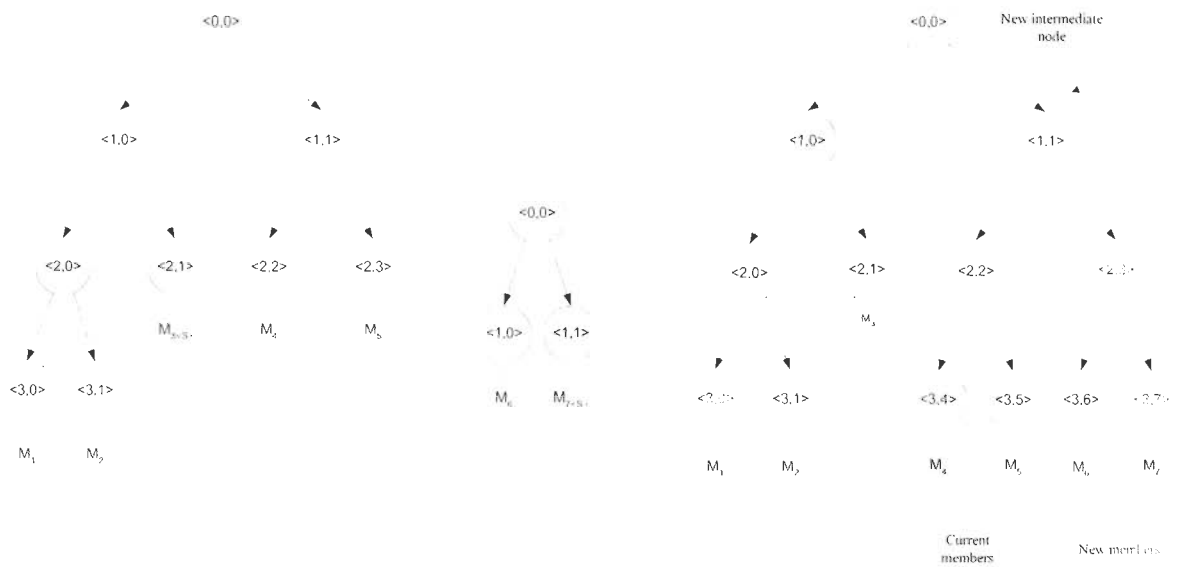


Figure 4.3: Merge Event in TGDH Protocol

All keys and blinded keys on the path from the merge point to the root node are invalidated. The rightmost member of the subtree rooted at the merge point becomes the sponsor of the key update operation. The sponsor computes all keys and blinded keys and broadcasts the tree with the blinded keys to all other members. All members now have the complete set of blinded keys, which allows them to compute all keys on their key path. Fig 4.3 illustrates merge event where members M_6 and M_7 are added to a group consisting of members M_1, M_2, M_3, M_4, and M_5. The merge event requires $2,3$ and $2\lceil \log_2 n \rceil$ number

of rounds, number of messages, and number of exponential operations respectively where n denotes the number of existing members in the group.

4.4.3. Single leave event.

Single member leave event is illustrated in Fig 4.4. Suppose that member M_5 leaves the system. Node $<3,4>$ is then promoted to node $<2,2>$, and nodes $<1,1>$ and $<0,0>$ become renewed nodes, defined as the non-leaf nodes whose associated keys in the key tree are renewed. Also, member M_4 becomes the sponsor. It renews the secret keys $K_{<1,1>}$ and $K_{<0,0>}$ and broadcasts the blinded keys $BK_{<1,1>}$ and $BK_{<2,2>}$ to all the members. Members M_2 and M_3, upon receiving the blinded key $BK_{<1,1>}$, compute the new group key $K_{<0,0>}$. Similarly, members M_6 and M_7, upon receiving $BK_{<2,2>}$, can compute $K_{<1,1>}$ and then the new group key $K_{<0,0>}$. The single leave event requires 1,1 and $\lceil \log_2 n \rceil$ number of rounds, number of messages, and number of exponential operations respectively where n denotes the number of existing members in the group.

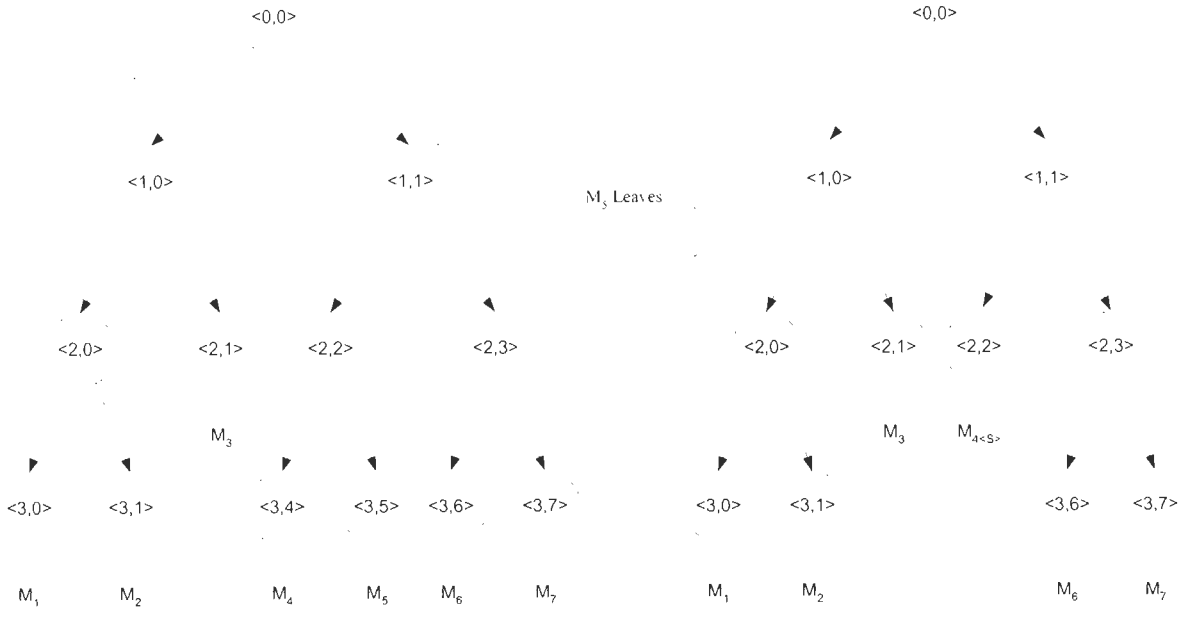


Figure 4.4: Single Leave Event in TGDH Protocol

4.4.4. Partition event.

In case of partitioning event, the protocol runs as follows: In the first round, each remaining member updates its view of the tree by deleting all leaf nodes associated with partitioned members and (recursively) their respective parent nodes.

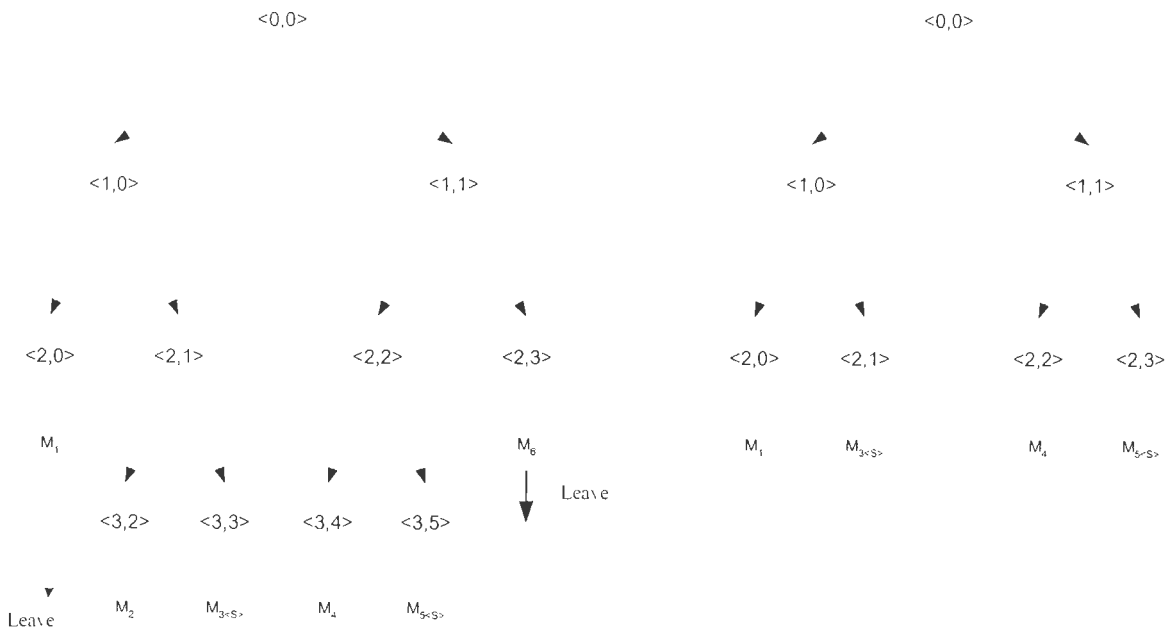


Figure 4.5: Partition Event in TGDH Protocol

To prevent re-use of an old group key, one of the remaining members changes its key share. To this end, in the first protocol round, the shallowest rightmost sponsor changes its share. Each sponsor then computes the keys and blinded keys as far up the tree as possible, and, then broadcasts the set of new blinded keys. Upon receiving the broadcast, each member checks whether the message contains a new blinded key. This procedure iterates until all members obtain the group key. Fig 4.5 illustrates a partition event, where members M_2 and M_6 are removed from the group. The partition event requires $\lceil \log_2 n \rceil, 2\lceil \log_2 n \rceil$ and $3\lceil \log_2 n \rceil$ number of rounds, number of messages, and number of exponential operations respectively where n denotes the number of existing members in the group.

4.5. The STR Protocol

The Skinny Tree (STR) protocol, proposed by steer et al. [28] and undertaken by Kim et al. in [114], is a contributive protocol using a tree structure. Fig 4.6 depicts an STR tree with three members.

The leaves are associated to group members. Each leaf is identified by its position LN_i in the tree and holds a secret random R_i (generated by the corresponding member M_i) and its public blinded version $BR_i = \alpha^{R_i} \bmod p$, where α and p are DH parameters. Each internal node is identified by its position IN_i in the tree and holds a secret random K_i and its blinded public version $BK_i = \alpha^{K_i} \bmod p$. Each secret K_i is recursively calculated as follows:

$$K_i = (BK_{i-1})^{R_i} \bmod p = (BR_i)^{K_{i-1}} \bmod p$$

The group key $K_n = \alpha^{R_n \alpha^{R_{n-1}} \ldots \alpha^{R_2 R_1}}$ is the key associated to the root. Due to the linear structure of the tree, this solution induces a O(n) key calculations in order to establish the group key associated to the root of the tree. Besides, each member should store and maintain all the public keys associated to all the nodes of the tree. In case of a membership change (join/leave) the tree is re-built consequently and hence all the members update the group key, which is the new key K_n associated to the root of the tree.

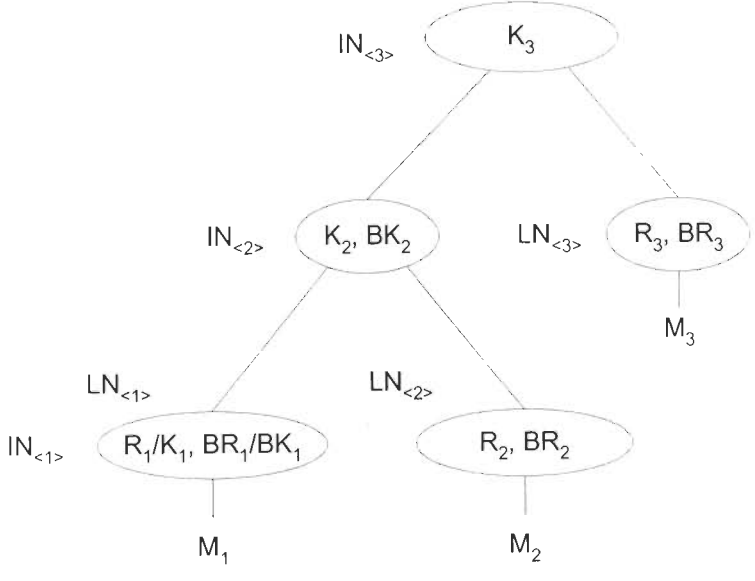


Figure 4.6: STR Key Tree with Three Members

4.5.1. Single join event.

The merge event requires $2,3$ and $3j$ number of rounds, number of messages, and number of exponential operations respectively where j denotes the number of joining members.

Fig 4.8 illustrates the merging two groups. After the merge notification, the sponsors M_3 and M_6 broadcast their key trees containing all blinded session randoms. Upon receiving these broadcast messages, every member in both groups reconstructs the key tree. Since the sponsor ids are used for grafting purpose in case of both the trees have same number of members. Therefore, The tree with sponsor M_6 (i.e., call it $T_{(1)}$) is placed above the tree with sponsor M_3 (i.e., call it $T_{(2)}$). Every member generates a new intermediate node $IN_{(4)}$ and makes it the parent of the old root node $IN_{(3)}$ of the tree $T_{(2)}$ and the previous leftmost leaf node $LN_{(4)}$. Both intermediate nodes $IN_{(1)}$ and $IN_{(2)}$ of tree $T_{(1)}$ then need to be renumbered as $IN_{(5)}$ and $IN_{(6)}$, respectively. The new intermediate node $IN_{(4)}$ also becomes the child of the previous lowest intermediate node $IN_{(5)}$. Using the previous blinded group key at $IN_{(3)}$ of the $T_{(2)}$ group and blinded session random BR_4 and BR_5, the sponsor in the second round, M_3, computes all intermediate secret and blinded keys ($K_3, BK_3, K_4, BK_4, K_5, BK_5$) except the root node. Finally, it broadcasts $BT_{(3)}$ that contains all blinded keys and blinded session randoms up to $IN_{(5)}$. Upon receipt of the broadcast, every member can compute the group key.

4.5.3. Single leave event.

We again have a group of n members when a member $M_l (l < n)$ leaves the group. If $l > 1$, the sponsor M_s is the leaf node directly below the leaving member, i.e., M_{l-1} otherwise, the sponsor is M_2. Upon hearing about the leave event from the group communication system, each remaining member updates its key tree by deleting the nodes $LN_{(l)}$ corresponding to M_l and its parent node $IN_{(l)}$.

The nodes above the leaving node are also renumbered. The former sibling $IN_{(l-1)}$ of M_l is promoted to replace (former) M_l's parent. The sponsor M_s selects a new secret session random, computes all secret as well as blinded keys just below the root node, and

broadcasts $BT_{(3)}$ to the group. This information allows all members (including the sponsor) to recompute the new group key.

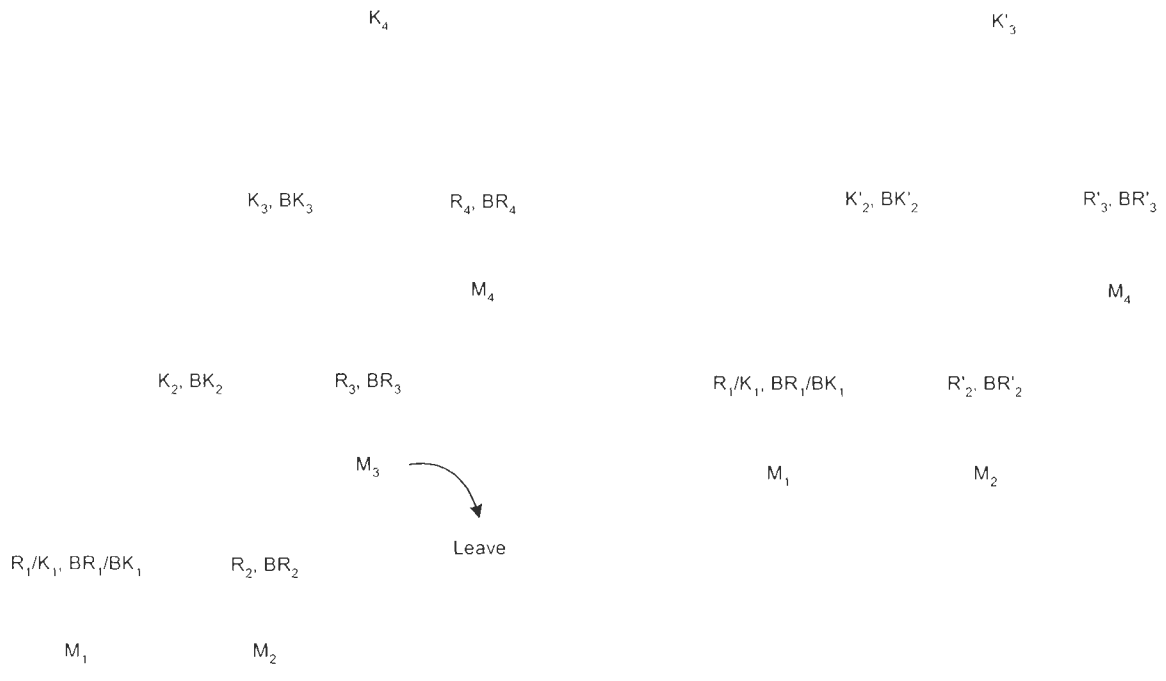


Figure 4.9: Single Leave Event in STR Protocol

Fig 4.9 illustrates that if member M_3 leaves the group, other members delete the leaving node along with its parent. Then, the sponsor M_2 picks its new session random R_2, computes $BR_2^{'}, K_2^{'} BK_2^{'}$, and broadcasts the updated tree $BT_{(3)}$. Upon receiving the broadcast, all members (including M_2) compute the group key K_3. Single leave event require one communication round and a single broadcast. The cryptographic cost varies depending upon two factors, first, the position of the departed member and second, the position of the remaining members needing to compute the new key. The single leave event requires 1,1 and $(3n/2+2)$ number of rounds, number of messages, and number of exponential operations respectively where n denotes the number of existing members in the group.

4.5.4. Partition event.

A network fault (RF link failure) or mobility of nodes can cause a partition of the existing group. To the remaining members, this actually appears as a concurrent leave of

multiple members. By introducing minor difference in sponsor selection in leave protocol, it can be used as partition protocol.

In case of a partition, the sponsor is the leaf node directly below the lowest-numbered leaving member. (If M_1 is the lowest-numbered leaving member, the sponsor is the lowest-numbered surviving member.). After deleting all leaving nodes, the sponsor M_s refreshes its session random (key share), computes secret and blinded keys going up the tree, as in the plain leave protocol, terminating with the computation of $\alpha^{K_{n-1}} \bmod p$. It then broadcasts the updated key tree $BT_{(s)}$ containing only blinded values. Each member (including M_s) can now compute the group key.

In Fig 4.10 the sponsor deletes all nodes of leaving members and computes all necessary secrets and blinded keys in the first round. Member M_1 is the sponsor since M_2 left the group. After picking a new session random R_1, the sponsor computes K_2 and $\alpha^{K_2} \bmod p$, and broadcasts the whole tree. Upon receiving this message, every member can compute the new group key K_3. And, session randoms and blinded session randoms are renumbered as in the leave protocol.

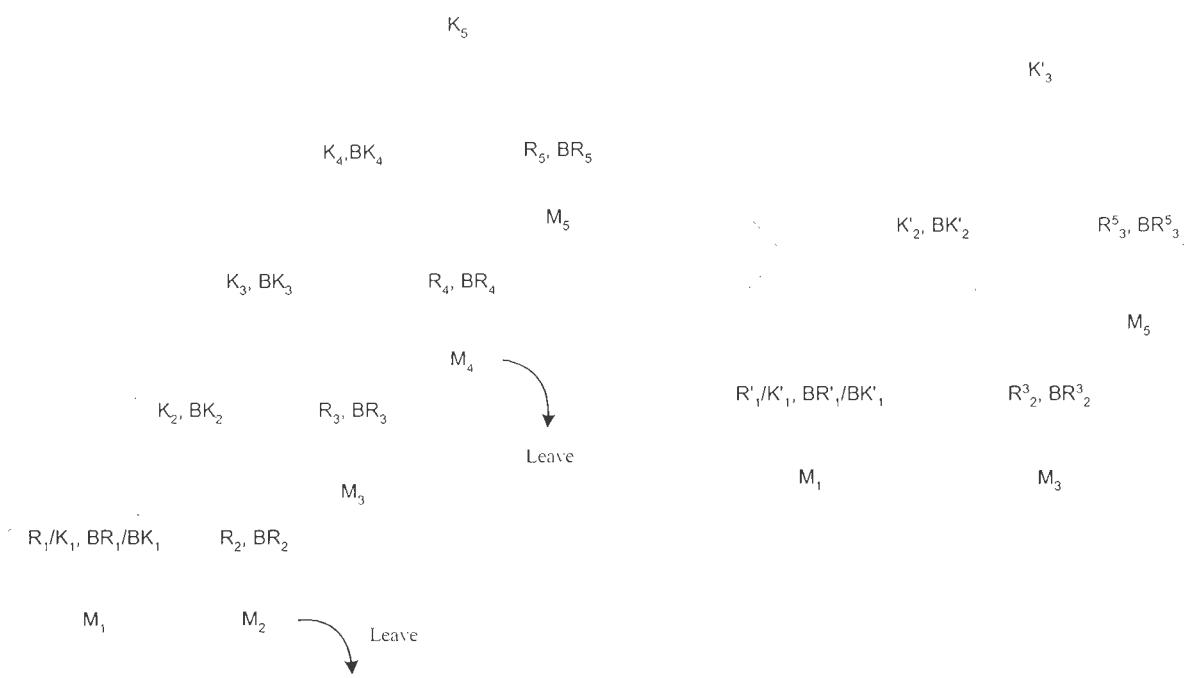


Figure 4.10: Partition Event in STR Protocol

The partition event requires 1,1 and $(3n/2+2)$ number of rounds, number of messages, and number of exponential operations respectively where n denotes the number of existing members in the group.

4.6. The NAGKA Protocol

Adrian Perrig [4] proposed a non-authenticated group key agreement (NAGKA) protocol. NAGKA protocol is a variation of the non-authenticated Diffie-Hellman two party key agreement protocol [110], extended to group agreement. The protocol works as follows: The key tree as shown in the Fig 4.11 is constructed from the leaves up to the root. Members are located at the leaf nodes.

Figure 4.11: Key tree (depth=2) in NAGKA Protocol

Initially each member chooses a random number for the key value of its leaf node. For the construction of keys at each level in key tree, two members establish a key through the

key agreement procedure, one member from the left subtree and the other from the right subtree. They both broadcast their part of the Diffie-Hellman key agreement, which allows all the members of the subtree to compute the key of the current level. Once the root key is established, the group can start the secure communication by encrypting all messages with the root key.

4.6.1. Single join event.

In this protocol, the new member always joins at the closest node to the root. As illustrated in Fig 4.12, the new member M_5 joins at node $<1,2>$ since it is the closest leaf node to the root. The member M_4 is shifted one level down to accommodate M_5. All keys from the new member up to the root such as $K_{<2,4>}$, $K_{<1,2>}$, and $K_{<0,1>}$ need to be renewed. The single join event requires $\lceil \log_2 n \rceil, \lceil \log_2 n \rceil + 1$ and $2\lceil \log_2 n \rceil$ number of rounds, number of messages, and number of exponential operations respectively where n denotes the number of existing members in the group.

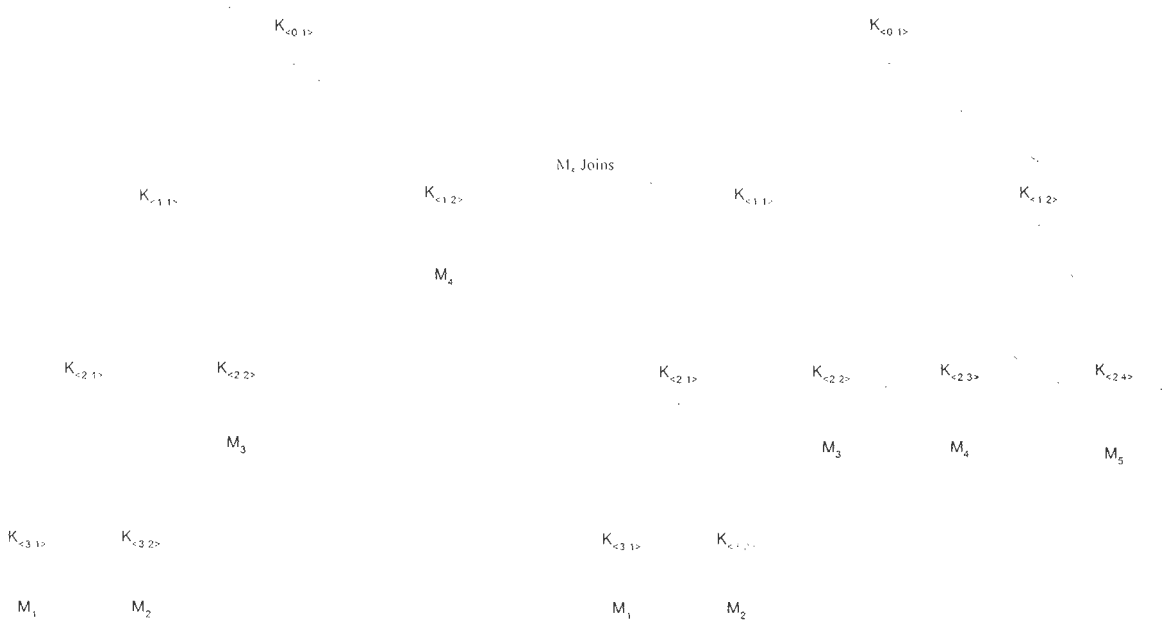


Figure 4.12: Single join event in NAGKA Protocol

4.6.2. Single leave event.

In this protocol, all the keys that leaving member knows are changed so that future communication among nodes is not available to this member.

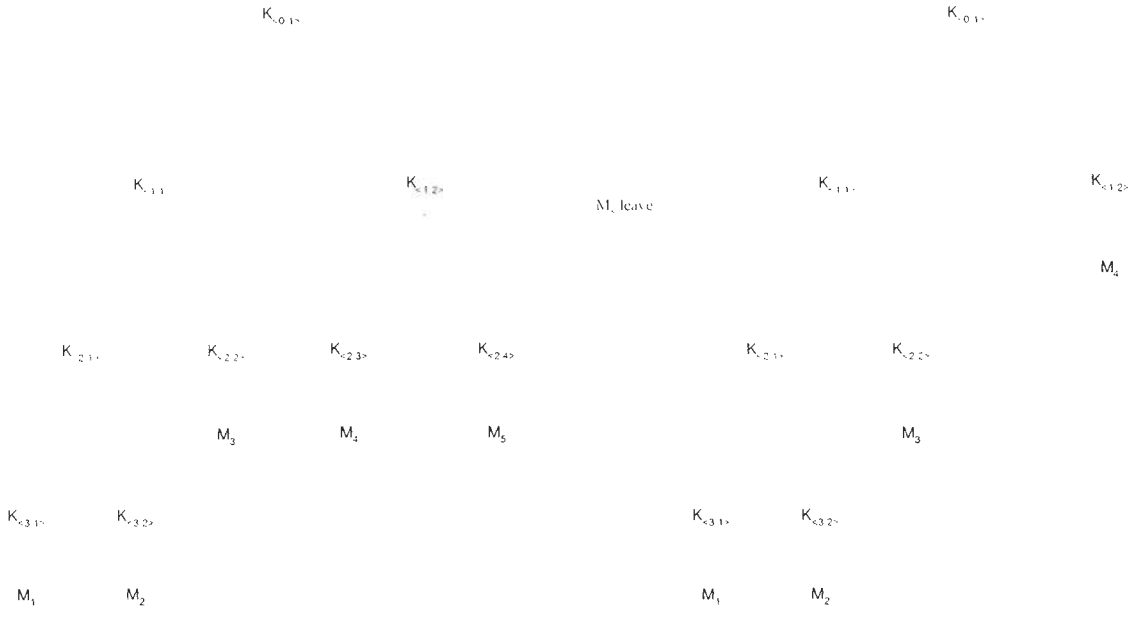


Figure 4.13: Single leave event in NAGKA Protocol

When the member at node $< h, i >$ leaves the group, the node $< h-1, \left\lceil \frac{i}{2} \right\rceil >$ is replaced by $< h, i \pm 1 >$ (by shifting the entire subtree one level up). All keys from the deleted node up to the root are updated i.e., When member M_5 leaves, then the node $< 1, 2 >$ is deleted and node $< 2, 3 >$ is shifted one level up, as illustrated in Fig 4.13. The single leave event requires $(\lceil \log_2 n \rceil - 1), (\lceil \log_2 n \rceil)$ and $2(\lceil \log_2 n \rceil - 1)$ number of rounds, number of messages, and number of exponential operations respectively where n denotes the number of existing members in the group.

4.7. Complexity Analysis

Group key agreement protocols, which enable the members to agree on a common secret value, based on public contribution of each member, do not require the presence of a central authority. Also, when the group composition changes due to join, leave, merge, and partition of member(s), one can employ supplementary key agreement protocols to get a new group key. Table 4.1 depicts a comparative analysis of various dynamic group key agreement protocols based on complexity parameters such as number of rounds, number of messages exchanged, and exponential operations required by these supplementary group key agreement

protocols, for rekeying operation in case of various events, i.e., join, leave, merge and partition event. The notations n, j and l denote current group members, joining member(s), and leaving member(s) respectively.

TABLE 4.1. COMPARISON OF DYNAMIC GROUP KEY AGREEMENT PROTOCOLS

Dynamic Group Key Agreement Protocol	Event	Number of Rounds	Number of Messages	Exponential operations	Round Synchronization
IKA.1	Join	2	2	$n+1$	No
	Merge	$j+1$	$j+1$	$\frac{1}{2}(j^2+2nj+j)$	
	Leave	1	1	$2n-1$	
	Partition	1	1	$2n-l$	
IKA.2	Join	4	$n+3$	$n+3$	No
	Merge	$j+3$	$n+2j+1$	$n+2j+1$	
	Leave	1	1	$n-1$	
	Partition	1	1	$n-l$	
TGDH	Join	2	3	$2.\lceil \log_2 n \rceil$	Yes
	Merge	2	3	$2.\lceil \log_2 n \rceil$	
	Leave	1	1	$\lceil \log_2 n \rceil$	
	Partition	$\lceil \log_2 n \rceil$	$2.\lceil \log_2 n \rceil$	$3.\lceil \log_2 n \rceil$	
STR	Join	1	2	2	Yes
	Merge	2	3	$3j$	
	Leave	1	1	$3n/2+2$	
	Partition	1	1	$3n/2+2$	
NAGKA	Join	$\lceil \log_2 n \rceil$	$\lceil \log_2 n \rceil+1$	$2.\lceil \log_2 n \rceil$	Yes
	Leave	$\lceil \log_2 n \rceil-1$	$\lceil \log_2 n \rceil$	$2.(\lceil \log_2 n \rceil-1)$	

The IKA.2 protocol is better than IKA.1 in case of leave and partition event and IKA.2 protocol needs small number of exponential operations as compared to IKA.1. However, IKA.2 is inferior to IKA.1 for join event. Since the Clique protocols do not need round synchronization, these protocols are an attractive option for mobile ad hoc networks. The cost for TGDH is the average value when the key tree is fully balanced. Member join is costly because TGDH protocol has to balance the key tree if it becomes unbalanced after join of member(s). It is suitable for member leave operation since it takes only one round and $\lceil \log_2 n \rceil$ modular exponentiation. However, partitioning is expensive which may cause more cascaded faults and long delays to agree on the group key. The partition/leave cost for STR

protocol is computed on average, since it depends on the depth of the lowest-numbered leaving member node. As observed from the Table 4.1, STR protocol is minimal in communication on every membership event. A.Perrig presented three group key agreement protocols in [4], i.e., NAGKA, AGKA, and AGKA-G. First protocol is non-authenticated group key agreement protocol. Second protocol the authenticated group key agreement protocol, wherein members authenticate each other using certificates issued by third parties. Third protocol is also authenticated version but it uses Gunther's identity based key agreement [14] also known as implicitly-certified key agreement. In these three protocols, Single member joining and leaving is described but no clear cut provision is made for merge and partition events. Therefore, these protocols are not suitable for mobile ad hoc networks where merging and partitioning events occur very frequently.

4.8. Conclusion

This chapter examined the various group key agreement protocols from the point of dynamic membership events, to assess their suitability for an ad hoc network. Although analytically STR protocol shows best results among the presented protocols, yet it is not suitable for an ad hoc network because of round synchronization required in various membership events.

Note, the protocols such as INGM, BD, Hypercube and Octopus discussed in previous chapter need to be executed afresh on each join, leave, merge and partition event. We also notice that the protocols such as IKA.1 and IKA.2 in CLIQUE Suite do not require round synchronization. And, routing of messages takes place on hop-to-hop basis and this is the basic characteristic of ad hoc network. Due to this reason, ad hoc networks are also known as multi-hop wireless ad hoc network. Therefore, Protocols included in CLIQUE Suite are well suited for dynamic membership events i.e., join, leave, merge and partition events.

Chapter 5

Proposed Protocol

5.1 Introduction

As observed from previous discussion, (Cf. Chapter 3 and 4), that among INGM [46], BD [71], Hypercube [12], Octopus [12], Clique (IKA.1 and IKA.2) [77], TGDH [113,115], NAGKA [4], and STR [114] protocols, the Clique protocols show the best results as far as setup operation and dynamic events (join, merge, leave and partition) are concerned. Although clique protocols show better results, yet these protocols are still expensive in terms of number of rounds, number of messages, and number of exponential operations. The number of rounds in clique protocols is directly proportional to group size which shows that these protocols are not scalable too. This Chapter proposes a new secure and more efficient group key agreement protocol for mobile ad hoc networks. The proposed protocol also has provisions for all valid members to detect errors and stop execution of the protocol immediately as they encounter invalid message from the corrupted members. The proposed protocol suite has a setup protocol which completes setup of a new group in two rounds. For authentication, each group member generates two signatures and performs $2n$ signature verifications. For dynamic membership events like join (merge), leave (partition), the *join-merge*, and *leave-partition* supplementary protocols have been presented, which also require two rounds for their successful completion. The *join-merge* and *leave-partition* supplementary protocols are executed to generate the new group key whenever some member(s) join (merge) or leave (partition) that may be voluntary or forced (in case of cheating) in a group.

5.2 The Model

This section presents a security model for a group key agreement protocol based on Bresson et al. [31,33] and Katz and Yung [49].

Participants: Participants are members of a nonempty set, G, which participate in the group key agreement protocol, P. Each member generates secret/public key pairs (sk, pk), and each member knows public keys of all members. The keys sk and pk are long-lived and are used for signature generation and verification respectively. An adversary is not a valid member of, G, but can hold all communication on a network and corrupt group members.

Partnering: Whenever group membership changes, a new group, $G_v = \{M_1, \ldots\ldots, M_n\}$, is formed and each group member of G_v can obtain a new group key gk_v through an instance performing, P [49]. $\prod_{M_i}^{j}$ denotes an instance j of a group member, M_i. The group index v increases whenever group membership changes and G_0 denotes the initial group. An instance, $\prod_{M_i}^{j}$, of a member, M_i, has unique session identifier $sid_{M_i}^{j}$ and partner identifier $pid_{M_i}^{j}$. After the group key agreement protocol, P, has been terminated successfully, $\prod_{M_i}^{j}$ has a unique session (group) key identifier $gk_{M_i}^{j}$ corresponding to the session (group) key gk_v. $pid_{M_i}^{j}$ corresponds to a set of group members $G_{M_i}^{j} = G_v \setminus \{M_i\}$, where symbol $\setminus$ stands for leave operation. When the group key agreement protocol, P, has successfully terminated in the instance, $\prod_{M_i}^{j}$, each member, M_k, of $G_{M_i}^{j}$ has an instance $\prod_{M_k}^{j_k} (1 \le k \ne i \le n)$ containing $\{sid_{M_k}^{j_k}, pid_{M_k}^{j_k}, gk_{M_k}^{j_k}\}$ such that $\{sid_{M_k}^{j_k} = sid_{M_k}^{j}\}$, $pid_{M_k}^{j_k} = G_v \setminus \{M_k\}$ and $gk_{M_k}^{j_k} = gk_{M_i}^{j}$. The instances $\prod_{M_i}^{j}$ and $\prod_{M_k}^{j_k}$ are partnered instances [49].

5.2.1 Protocol Model

The group key agreement protocol, P, consists of the following algorithm:

Key generation: This probabilistic time algorithm produces long-lived keys for each member of G when an input value 1^{sp} where sp is a security parameter is provided.

Setup(G_0): This algorithm helps in commencement of group key agreement protocol, P, and generates the initial session (group) key gk_0.

Join-Merge (J, G_{v-1}): Input to this algorithm is a set of joining members' identities denoted by J and the current group, G_{v-1}. The output of this algorithm is a new group $G_v = G_{v-1} \bigcup J$,

where symbol $\cup$ stands for join operation. All members of G_v share a new session (group) key gk_v secretly.

Leave-Partition (L, G_{v-1}): Input of this algorithm is a set of leaving members' identities denoted by L and the current group, G_{v-1}. The output of this algorithm is a new group, $G_v = G_{v-1} \setminus L$, where symbol $\setminus$ for leave operation. All members of G_v share a new session (group) key gk_v secretly.

5.2.2 Security Model

The security model defines the capabilities of an adversary, A. It allows the adversary, A, to potentially hold all communication in the network via access to a set of oracles as defined below. The model considers an experiment in which the adversary, A, asks queries to oracles, and the oracles answer back to the adversary, A. Oracle queries model attacks which an adversary, A, may use in the real system [67]. The model considers the following types of queries in this work.

Send$(\prod_{M_i}^{j}, m)$: An adversary, A, sends a message m to an instance, $\prod_{M_i}^{j}$. When $\prod_{M_i}^{j}$ receives m, it responds according to the group key agreement protocol. An adversary, A, may use this query to perform active attacks by modifying and inserting the messages of the key agreement protocol. Impersonation attacks and bucket-brigade attack are also possible using this query.

Setup(G_0), **Join-Merge** (J, G_{v-1}), **Leave-Partition** (L, G_{v-1}): Using these queries , an adversary, A, can commence the *setup*, *join-merge* or *leave-partition* algorithm.

Reveal$(\prod_{M_i}^{j})$: An adversary, A, can obtain a session (group) key gk which has been exchanged between instance, $\prod_{M_i}^{j}$, and partnered instances whereas M_i's long-lived key are concealed. The query models known key attacks (or Denning-sacco attacks [23]).

Corrupt(M_i): An adversary, A, can obtain M_i's long-lived key. In our protocol, we consider adaptive corruptions [109]; in general, adaptive corruptions mean weak corruptions in which an adversary can obtain an honest members' long-lived key, but cannot obtain the members' ephemeral keys.

Test($\prod_{M_i}^{j}$): This query is used to define the advantage of an adversary. An adversary, A, executes this query on a fresh instance, $\prod_{M_i}^{j}$, at any time, but only once (other queries have no restriction). When an adversary, A, asks this query, it receives a session (group) key gk of the instance $\prod_{M_i}^{j}$, if $b=1$ or a random string if $b=0$ where b is the result of a coin-flip, finally, an adversary, A, produces a bit b'.

To define a meaningful notion of security, we first define freshness.

Definition 1. An instance, $\prod_{M_i}^{j}$, fresh if both the following conditions are true at the end of the experiment described above:

(i) None of the instance, $\prod_{M_i}^{j}$, and its partnered instances has received an adversary's Reveal query.

(ii) No one of M_i and other members in $G_{M_i}^{j}$ has received an adversary's Corrupt query before adversary's Send queries.

Let P be a group key agreement protocol and let A be an active adversary against P. When A asks a Test query to a fresh instance, $\prod_{M_i}^{j}$, in P, A receives the result of the coin flip b, which is either a session (group) key or a random value and then output a bit b'. If the probability that A correctly guesses the bit b is negligible. P is secure in the sense that A cannot obtain any information about a session (group) key through re-keying broadcast messages. Let $Adv_{P,A}^{gka}$ denote the advantage for A's guess over the result of a coin-flip in a Test query with P. Then, $Adv_{P,A}^{gka}$ is defined as follows:

$$Adv_{P,A}^{gka} = \Pr[b' = 1 \mid b = 1] - \Pr[b' = 1 \mid b = 0] = 2\Pr[b' = b] - 1$$

It can be said that P is a secure group key agreement (gka) protocol if $Adv_{P}^{gka} = \max_{A}\{Adv_{P,A}^{gka}\}$ is negligible. For the security of authentication, the model considers the ability of A for impersonation attacks against a group member, M_i, in an instance, $\prod_{M_i}^{j}$ [33]. For impersonation attacks, A should be able to forge a signature of the group member, M_i, in the instance, $\prod_{M_i}^{j}$. If it is computationally infeasible that A generates a valid signature with any message under a chosen message attack, It can be said that the signature scheme is CMA-secure. Let $\sum = (K, S, V)$ be a signature scheme where K, S and

V are key generation, signing and verification algorithms. Formally, let $Succ_{\Sigma,A}^{cma}$ be a success probability of A's existential forgery under a chosen message attack against Σ. Then, it may be stated that Σ is CMA-secure [26], if $Succ_{\Sigma}^{cma} = \max_{A}\{Succ_{\Sigma,A}^{cma}\}$ is negligible.

Let $G = \langle \alpha \rangle$ be a group. Given α^x and α^y, Computational Diffie-Hellman (CDH) problem is to compute a value α^{xy} [110]. For the CDH problem, Consider a probability $Succ_{G}^{cdh}$ such that

$$Succ_{G,A}^{cdh} = \Pr[C = \alpha^{xy} \mid \alpha^x, \alpha^y \leftarrow G; C \leftarrow A(\alpha^x, \alpha^y)],$$
$$Succ_{G}^{cdh} = \max_{A}\{Succ_{G,A}^{cdh}\}$$

5.3 Proposed Protocol Suite

This section presents the proposed protocol suite, which is a natural extension of two party Diffie-Hellman protocol [110]. And based on a secure signature scheme $\Sigma = (K, S, V)$. A group key space belongs to $\{0,1\}^{sp}$ where sp is a security parameter. Let $G = \langle \alpha \rangle$ be a cyclic group of prime order p. The parameters α and p are public parameters and $sp \leq |p|$ is satisfied. Let $H : \{0,1\}^* \rightarrow \{0,1\}^{sp}$ be a one-way hash function.

5.3.1 Key generation

Each member M_i of G_0 has a private/public key pair (sk_{M_i}, pk_{M_i}) for signing/verifying. The list of public keys is published to all members.

5.3.2 Setup Algorithm

Let $G_0 = \{M_1, ..., M_n\}$ be an initial group. The members of G_0 form a logical ring structure and members' indices could be considered on the circulation of $\{1, ..., n\}$. $L(i)$ and $R(i)$ means the left and right index of i^{th} member on the ring for $i \in \{1, ..., n\}$. Let $I_0 = ID_{M_1} \| ... \| ID_{M_n}$. Fig 5.1 shows the details of this algorithm with four members.

Round 1: Each member M_i randomly chooses $k_i \in \{0,1\}^{sp}$ and $x_i \in Z_p^*$, computes $y_i = \alpha^{x_i}$ and keeps k_i secretly. The last member, M_n, computes $H(k_n \| 0)$. Each member M_i

generates a signature $\rho_i^1 = S_{sk_{M_i}}(m_i^1 \| I_0 \| 0)$ where $m_i^1 = y_i$ for $1 \le i \le n-1$ and $m_n^1 = H(k_n \| 0) \| y_n$, and broadcasts $m_i^1 \| \rho_i^1$.

Round 2: All members receive $m_i^1 \| \rho_i^1$'s and verify ρ_i^1's. If some signature is not valid, this means, ρ_i^1 is signed by an adversary. In this case the signature verification, using member's public key, fails and the protocol halts. Otherwise, M_i computes $t_i^L = H(y_{L(i)}^{x_i} \| I_0 \| 0), t_i^R = H(y_{R(i)}^{x_i} \| I_0 \| 0)$ and generates $T_i = t_i^L \oplus t_i^R$. The last member M_n additionally computes $\hat{T} = k_n \oplus t_n^R$. Each member M_i generates $\rho_i^2 = S_{sk_{M_i}}(m_i^2 \| I_0 \| 0)$ and broadcast it, where $m_i^2 = k_i \| T_i$ for $1 \le i \le n-1$ and $m_n^2 = \hat{T} \| T_n$.

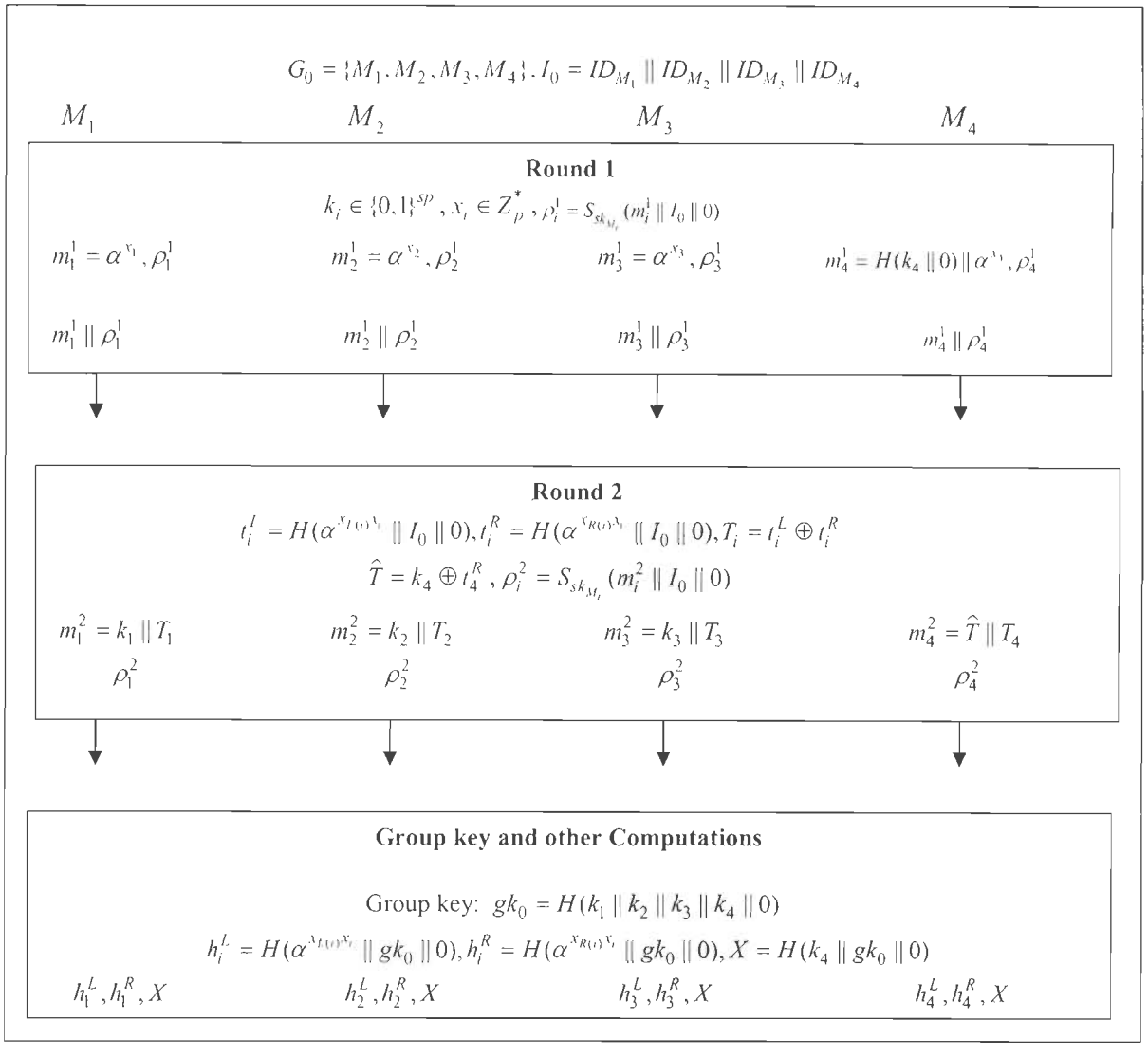


Figure 5.1: Setup algorithm with $G_0 = \{M_1, M_2, M_3, M_4\}$

Group key and other computations: All members verify signatures ρ_i^2's. If all signatures are valid, M_i computes $\tilde{t}_{i+1}^R, \tilde{t}_{i+2}^R, ..., \tilde{t}_{i+(n-1)}^R (= \tilde{t}_i^L)$ by using t_i^R:

$$\tilde{t}_{i+1}^R = T_{i+1} \oplus t_i^R, \tilde{t}_{i+2}^R = T_{i+2} \oplus t_{i+1}^R, ..., \tilde{t}_{i+(n-1)}^R = T_{i+(n-1)} \oplus \tilde{t}_{i+(n-2)}^R$$

Finally M_i can check if $t_i^L = \tilde{t}_i^L$ holds, for example, if there are four members (M_1, M_2, M_3, M_4) in the group, then, for member, M_2, the value $(((T_1 \oplus T_3 \oplus T_4) \oplus t_{2,3})$ should be equal to $t_{1,2}$. Otherwise, message validity check fails and protocol halts. Even though illegitimate members or system faults broadcast wrong messages (or no message), yet honest members can notice the errors through the above check process and then halt the protocol. However, it is not easy to find who transmitted illegal messages. When members want to find illegitimate members, all members participating in this protocol should reveal their secret values x_i's. If the above check process has been valid, all members have $\tilde{t}_n^R (= t_n^R)$. Then they can obtain k_n from $\hat{T}$ and check if $H(k_n \| 0) = \hat{k}_n$ holds. This check ensures the control of key and the one-way hash function H. Each member computes a group key as $gk_0 = H(k_1 \| k_2 \| ... \| k_{n-1} \| k_n \| 0)$, in addition to, $h_i^L = H(y_{L(i)}^{x_i} \| gk_0 \| 0), h_i^R = H(y_{R(i)}^{x_i} \| gk_0 \| 0)$, $X = H(k_n \| gk_0 \| 0)$ and saves (h_i^L, h_i^R, X, gk_0) secretly. And all members erase other ephemeral data.

5.3.3 Join-Merge Algorithm

Let $G_{v-1} = \{M_1, ..., M_n\} (v \geq 1)$ be the current group and $J = \{M_{n+1}, ..., M_{n+n'}\} (n' \geq 1)$ be a set of newly joining members. The algorithm divides G_{v-1} into three parts $\{M_1\}, \{M_2, ..., M_{n-1}\}$ and $\{M_n\}$, and consider M_2 as an agent of $\{M_2, ..., M_{n-1}\}$. For simplicity, the member (s) $M_{n+n'+1}, M_{n+n'+2}$ and $M_{n+n'+3}$ denote $M_1, (M_2, ...M_{n-1})$, and M_n. In this algorithm, we consider a ring structure among the members $M_{n+1}, ..., M_{n+n'+3}$. Let G be the set $\{M_{n+1}, ..., M_{n+n'+3}\}$ and $I_v = ID_{M_1} \| ... \| ID_{M_{n+n'+3}}$ Fig 5.2 shows the details of the Join-Merge protocol.

Round 1: Each member M_{n+i} of G randomly chooses $k_{n+i} \in \{0,1\}^{sp}$ and $x_{n+i} \in Z_p^*$, computes $y_{n+i} = \alpha^{x_{n+i}}$ and keeps k_{n+i} secretly. The member $M_{n+n'+2}(= M_2)$ computes $y_{n+n'+2} = \alpha^X$ by using the secret value X instead of $x_{n+n'+2}$ and the member $M_{n+n'+3}(= M_n)$ computes $H(k_{n+n'+3} \| v)$. Each member M_{n+i} generates $\rho_{n+i}^1 = S_{sk_{M_{n+i}}}(m_{n+i}^1 \| I_v \| v)$ where $m_{n+i}^1 = y_{n+i}$ for $1 \le i \le n'+2$ and $m_{n+n'+3}^1 = H(k_{n+n'+3} \| v) \| y_{n+n'+3}$, and broadcasts $m_{n+i}^1 \| \rho_{n+i}^1$.

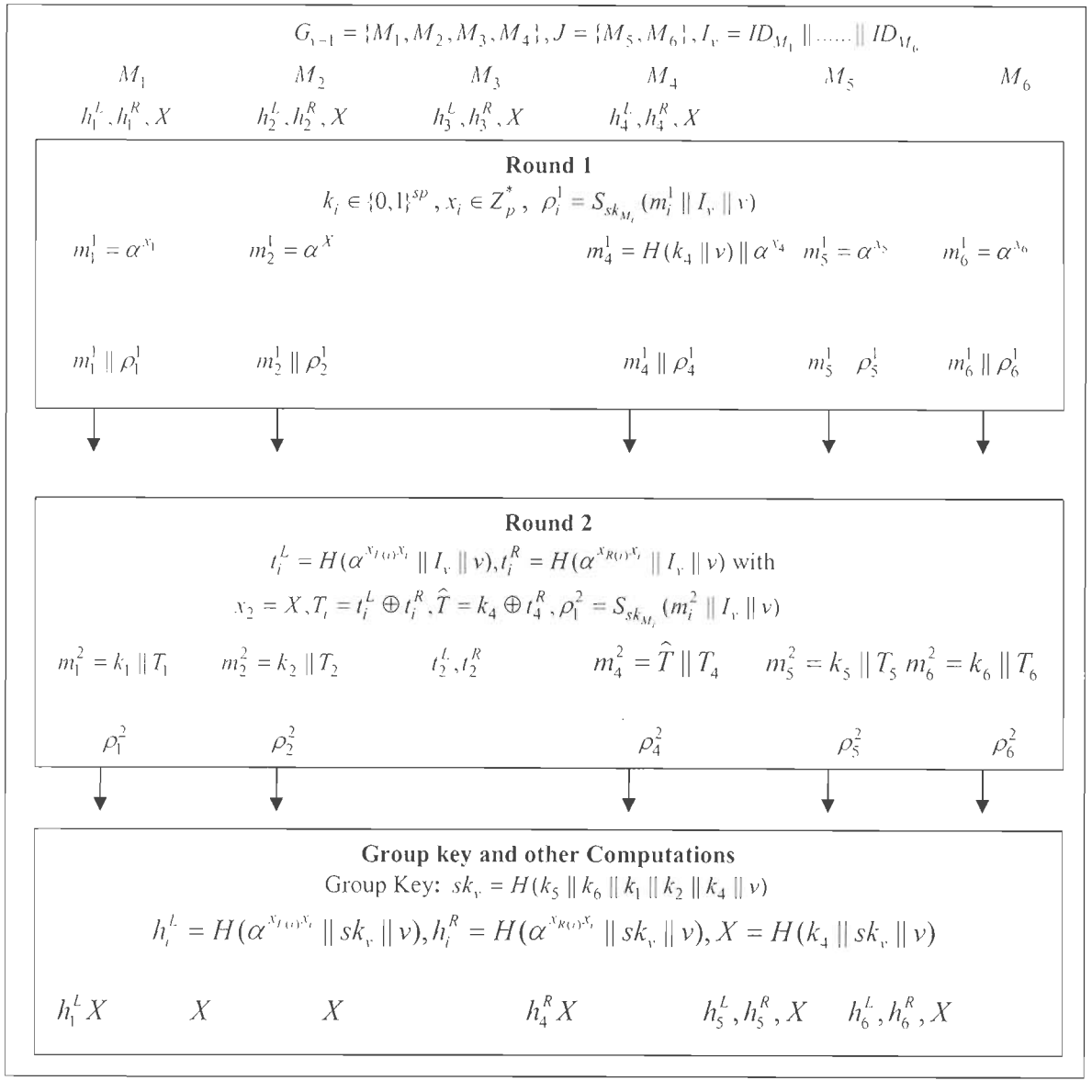


Figure 5.2: Join-Merge algorithm with $G_{v-1} = \{M_1, M_2, M_3, M_4\}, J = \{M_5, M_6\}$

Round 2: All members receive $m_{n+i}^1 \| \rho_{n+i}^1$'s and verify ρ_{n+i}^1's. If some signature is not valid, this means, ρ_{n+i}^1 is signed by an adversary. In this case the signature verification, using

member's public key, fails and the protocol halts. Each member M_{n+i} computes $t^L_{n+i} = H(y^{x_{n+i}}_{L(n+i)} \| I_v \| v), t^R_{n+i} = H(y^{x_{n+i}}_{R(n+i)} \| I_v \| v)$ and generates $T_{n+i} = t^L_{n+i} \oplus t^R_{n+i}$. The member, $M_{n+n'+3}$, additionally computes $\hat{T} = k_{n+n'+3} \oplus t^R_{n+n'+3}$. Each member M_{n+i} generates $\rho^2_{n+i} = S_{sk_{M_{n+i}}}(m^2_{n+i} \| I_v \| v)$ and broadcasts ρ^2_{n+i} where $m^2_{n+i} = k_{n+i} \| T_{n+i}$ for $1 \le i \le n'+2$ and $m^2_{n+n'+3} = \hat{T} \| T_{n+n'+3}$. All members $\{M_3, ..., M_{n-1}\}$ compute $t^L_{n+n'+2}$ and $t^R_{n+n'+2}$ by using X.

Group key and other computations: All members verify ρ^2_{n+i}'s. If all signatures are valid, each member M_{n+i} computes $\tilde{t}^R_{n+i+1}, ..., \tilde{t}^R_{n+i+(n-1)} (= \tilde{t}^L_{n+i})$ by using t^R_{n+i} and checks if $t^L_{n+i} = \tilde{t}^L_{n+i}$ holds. Also, the members $\{M_3, ..., M_{n-1}\}$ can check it by using $t^L_{n+n'+2}$ and $t^R_{n+n'+2}$. Finally all members can obtain $k_{n+n'+3}$ from $\hat{T}$ and compute a new group key gk_v as follows:

$$gk_v = H(k_{n+1} \| ... \| k_{n+n'+3} \| v).$$

Each new member $M_{n+i} (1 \le i \le n')$ generates $h^L_{n+i} = H(y^{x_{n+i}}_{L(n+i)} \| gk_v \| v)$ and $h^R_{n+i} = H(y^{x_{n+i}}_{R(n+i)} \| gk_v \| v)$. M_1 and M_n respectively compute $h^L_1 = H(y^{x_1}_{n+n'} \| gk_v \| v)$ and $h^R_n = H(y^{x_n}_{n+1} \| gk_v \| v)$ instead of the previous value $h^L_1 (= h^R_n)$. All members compute a new value $X = H(k_n \| gk_v \| v)$. Each member M_i saves h^L_i, h^R_i, X and gk_v secretly.

5.3.4 Leave-Partition Algorithm

Let $G_{v-1} = \{M_1, ..., M_n\}$ be the current group and $L = \{M_{l_1}, M_{l_2}, ..., M_{l_{n'}}\}$ with $\{l_1, ..., l_{n'}\} \subset \{1, 2, ..., n\}$ be a set of leaving members. Let $N(L)$ be a set of all left/right members of leaving members, *i.e.*, $N(L) = \{M_{l_1-1}, M_{l_1+1}, ..., M_{l_{n'}-1}, M_{l_{n'}+1}\}$.

For generating a new group $G_v = G_{v-1} \setminus L$ with a new group key gk_v, a new Diffie-Hellman value should be shared between two members M_{l_j-1} and $M_{l_j+1} (1 \le j \le n'')$. The members of G_v are considered as in ring structure and new members are indexed as $G_v = \{M_1, M_2, ..., M_{n-n'}\}$. Let $I_v = ID_{M_1} \| ... \| ID_{M_{n-n'}}$. Fig 5.3 shows the details of this protocol.

Round 1: Each member, M_w, of $N(L)$ randomly chooses $k_w \in \{0,1\}^{sp}$ and $x_w \in Z_p^*$, computes $y_w = \alpha^{x_w}$ and keeps k_w secretly. The member, $M_{l_{u-1}+1}$, computes $H(k_{l_{u-1}+1} \| v)$. The member, M_w, generates $\rho_w^1 = S_{sk_{M_w}}(m_w^1 \| I_v \| v)$ where $m_w^1 = y_w$ with $w \in \{l_1 - 1, l_1 + 1, \ldots, l_u - 1\}$ and $m_{l_{u-1}+1}^1 = H(k_{l_{u-1}+1} \| v) \| y_{l_{u-1}+1}$, and broadcasts $m_w^1 \| \rho_w^1$.

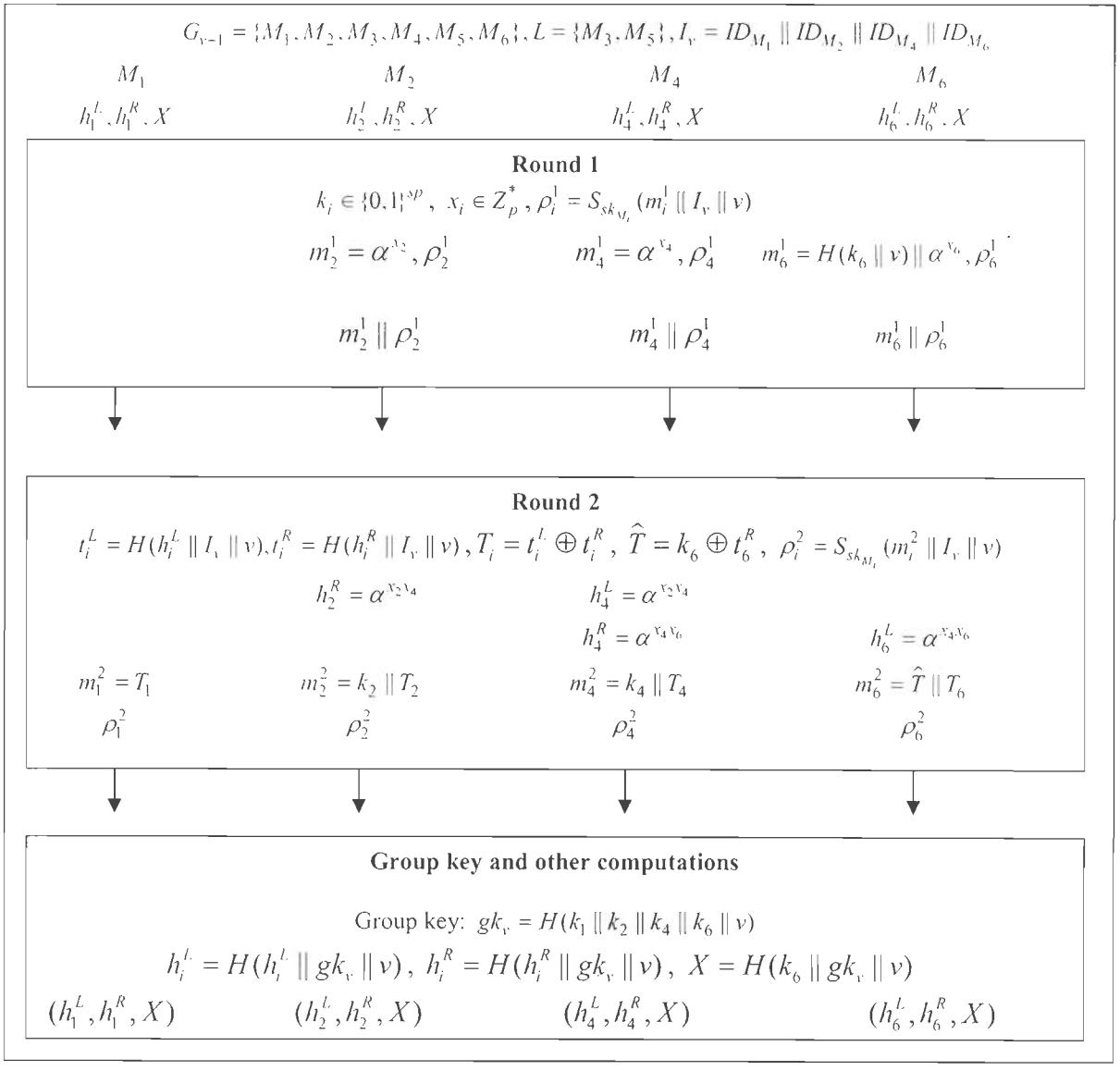


Figure 5.3: Leave-Partition algorithm with $G_{v-1} = \{M_1, M_2, M_3, M_4, M_5, M_6\}$ and $L = \{M_3, M_5\}$

Round 2: All members of G_v verify signatures ρ_w^1's. If some signature is not valid, this means, ρ_w^1 is signed by an adversary. In this case the signature verification, using member's

public key, fails and the protocol halts. If all signatures are valid then each member, $M_{l_{n'-1}}$ (resp. $M_{l_{n'+1}}$), of $N(L)$ regenerates $h^R_{l_{n'-1}} = y^{x_{l_{n'-1}}}_{l_{n'+1}}$ (resp. $h^L_{l_{n'+1}} = y^{x_{l_{n'+1}}}_{l_{n'-1}}$). And, each member, M_i, of G_v computes $t^L_i = H(h^L_i \| I_v \| v), t^R_i = H(h^R_i \| I_v \| v)$ and $T_i = t^L_i \oplus t^R_i$. The member, $M_{l_{n'+1}}$, additionally computes $\hat{T} = k_{l_{n'+1}} \oplus t^R_{l_{n'-1}}$. Each member, M_i, generates a signature $\rho^2_i = S_{sk_{M_i}}(m^2_i \| I_v \| v)$ and broadcasts ρ^2_i where $m^2_{l_{n'+1}} = \hat{T} \| T_{l_{n'+1}}, m^2_i = k_i \| T_i$ for other members except $M_{l_{n'+1}}$ of $N(L)$ and $m^2_i = T_i$ for members of $G_v \setminus N(L)$.

Group key and other computations: All members verify signatures ρ^2_i's. If all signatures are valid, each member, M_i, computes $\tilde{t}^R_{i+1}, \tilde{t}^R_{i+2}, \ldots, \tilde{t}^R_{i+(n-n'-1)} (= \tilde{t}^L_i)$ by using t^R_i. Finally, each member, M_i, checks if $t^L_i = \tilde{t}^L_i$ holds. Then all members compute a group key as follows:

$$gk_v = H(k_1 \| \ldots \| k_{l_1-2} \| k_{l_1-1} \| k_{l_1+1} \| \ldots \| k_{l_{n'-1}} \| k_{l_{n'+1}} \| v)$$

And, Each member, M_i, regenerates $h^L_i = H(h^L_i \| gk_v \| v)$, $h^R_i = H(h^R_i \| gk_v \| v)$ and $X = H(k_{l_{n'+1}} \| gk_v \| v)$ and saves h^L_i, h^R_i, X and group key gk_v.

5.4 Security Analysis

This section proves the security of the proposed protocol in the random oracle model [67]. The security of proposed protocol P is dependent on the probabilities $succ^{cma}_{\Sigma}$ and $succ^{cdh}_{G}$, since an adversary A against P can obtain information about group key only by two methods. An adversary, A, successfully performs either signature forgery attacks or CDH attacks. Even if random values k_i's were selected identically in different instances, A could not get any information about group key because of the index v and the random hash oracle H.

Theorem 5.1. *Let A be an active adversary against our protocol P in the random oracle model. Let q_S be the number of Send queries and q_H be the number of queries to the hash oracle H. Then,*

$$Adv^{gka}_P \le 2n.Succ^{cma}_{\Sigma}(t, q_S) + 2q_H q^2_S.Succ^{cdh}_G(t)$$

where n is the maximum number of group members and t is the adversary's running time.

Proof. The analysis considers $A's$ attacks as a sequence of simulated protocols, which is denoted by a sequence of experiments $\{Exp_0, ..., Exp_3\}$. In each experiment, Adversary, A, executes *Test* query and get a result of a coin flip b. Each $Succ_i$ denotes an event in which $A's$ guessing bit b' is equal to b in each Exp_i. Each Exp_i is simulated as follows:

Exp_0: This experiment is equal to the real protocol P. All group members obtain a pair of valid signing/verifying key and randomly choose $k_i's$ and $x_i's$. In this game, $A's$ advantage is equal to the advantage in the real protocol P. Thus,

$$\Pr[Succ_0] = \frac{Adv_{P,A}^{gka} + 1}{2} \qquad (1)$$

Exp_1: In this experiment, the analysis considers a special event *SigForge* wherein A executes a Send query with a message m instead of a group member M_i in an instance $\prod_{M_i}^{j}$ and the message is verified and accepted by all group members. In particular, the message m previously has not been used in any instances and a $Corrupt(M_i)$ query has not been executed to the member, M_i. When the event *SigForge* occurs, this experiment halts and $A's$ output b' is determined randomly. The difference between $A's$ outputs in experiments Exp_0 and Exp_1 is dependent on the event, *SigForge*. That is,

$$\left|\Pr[Succ_1] - \Pr[Succ_0]\right| \le \Pr[SigForge].$$

If one correctly guesses a member impersonated by A and the event *SigForge* occurs to the member, one can be successful in the existential forgery against a pair of signing/verifying key under CMA. Therefore we know that

$$Succ_{\Sigma,A}^{cma}(t, q_S) \ge \frac{1}{n} \Pr[SigForge]$$

Finally, we get

$$\left|\Pr[Succ_1] - \Pr[Succ_0]\right| \le \Pr[SigForge] \le n.Succ_{\Sigma,A}^{cma}(t, q_S) \qquad (2)$$

Exp_2: In this experiment, a Diffie-Hellman triple $(A = \alpha^a, B = \alpha^b, C = \alpha^{ab})$ is given. Whenever two successive members M_i and M_{i+1} should choose random values x_i and x_{i+1} and compute $y_i = \alpha^{x_i}$ and $y_{i+1} = \alpha^{x_{i+1}}$, the analysis simulates this experiment with $y_i = A^{c_i}$ and $y_{i+1} = B^{c_{i+1}}$ where c_i and c_{i+1} are random values in Z_p^*. Then a hash value $t_i^R (= t_{i+1}^L)$ is

computed by using $C^{c_i c_{i+1}}$. We know that this experiment is equal to Exp_1 as long as c_i and c_{i+1} are selected randomly. Therefore,

$$\Pr[Succ_2] = \Pr[Succ_1] \tag{3}$$

Exp_3 : In this experiment, a pair $(A = \alpha^a, B = \alpha^b)$ is given and there is no information about the Diffie-Hellman value $C = \alpha^{ab}$. Whenever two successive members M_i and M_{i+1} should choose random values x_i and x_{i+1} and compute y_i and y_{i+1}, the analysis simulates this experiment like Exp_2. However, when M_i and M_{i+1} should broadcast a message with a hash value $t_i^R (= t_{i+1}^L)$, a random value r in $\{0,1\}^{sp}$ is used as the hash value. Now, analysis considers an event Hash wherein A detects the fact that the broadcasted hash value t_i^R (or t_{i+1}^L) is incorrect by using $A's$ hash oracle queries. This event is possible when A sends a correctly guessing value $C^{c_i c_{i+1}}$ to the hash oracle H and receives a hash value. At that time, A recognizes that the value is different from the previous random value, r. When the event Hash occurs, this experiment is halted and $A's$ output b' is randomly chosen. Therefore,

$$\left|\Pr[Succ_3] - \Pr[Succ_2]\right| \le \Pr[Hash].$$

Given (A,B) one can obtain a valid Diffie-Hellman value C if both of the following situations occur; (1) two successive members compute $y_i = A^{c_i}$ and $y_{i+1} = B^{c_{i+1}}$ and use a random value r as a hash value t_i^R, (2) A executes a hash oracle query with a correctly guessing value $C^{c_i c_{i+1}}$ after (1), *i.e.*, the event Hash occurs. Therefore,

$$Succ_{G,A}^{cdh}(t) \ge \frac{1}{q_H q_S^2} \Pr[Hash]$$

And finally got

$$\left|\Pr[Succ_3] - \Pr[Succ_2]\right| \le \Pr[Hash] \le q_H q_S^2 . Succ_{G,A}^{cdh}(t) \tag{4}$$

Furthermore, A has no advantage for guessing a coin-flip bit b in this experiment since the hash oracle H has been supposed the random oracle and each input of the hash oracle is used only once owing to the index v. Therefore, $\Pr[Succ_3] = \frac{1}{2}$, hence theorem is proved.

The secure signature scheme ($\sum$) and hash function (H) may be replaced by ElGamal digital signature algorithm [106] and secure hash algorithm (SHA-1) [24] respectively for

practical implementation. During member authentication in second round and after second round of each algorithm, the messages are verified, if any impersonation attack has been done that may get detected due to selected pseudonumber, which is created in addition to secret key by each member in key generation phase.

5.5 Complexity Analysis

Table 5.1 shows the comparison between the proposed protocol and clique protocol suite (IKA.1 and IKA.2) [77]. For comparison, the efficiency measures, such as, number of rounds, number of messages and number of exponential operations have been considered. The notations n, j and l denote current group members, joining member(s), and leaving member(s) respectively. The proposed protocol suite does not need round synchronization. Thereby, no synchronous mechanism is needed. The number of rounds in setup protocol of IKA.1 and IKA.2 measures as n and $n+1$ (where n is the number of group members) whereas our protocol needs only two rounds irrespective of the group size as shown in Fig 5.4. *join-merge* (for join or merge) and *leave-partition* (for leave or partition) protocols also needs two rounds each for their completion, which indicates that the proposed protocol suite is scalable too. The setup protocol in proposed protocol suite needs $3n$ exponential operations whereas IKA.1 and IKA.2 need $(\frac{n}{2}(n+3)-1)$ and $(5n-6)$ exponential operations respectively as shown in Fig 5.5. However, the number of messages n in IKA.1 protocol is less as compared to IKA.2 and the proposed protocol, which need $2n-1$ and $2n$ messages respectively as shown in Fig 5.6. It is also seen that the *join-merge* of our protocol is more efficient in number of exponential operations measure as compared to join (merge) protocol of IKA.1 and IKA.2 protocols for large group size. And, the proposed *leave-partition* protocol is also efficient as compared to IKA.1 and IKA.2 when the n (the number of current group members) is high and l (the leaving members) is low. Our *leave-partition* protocol takes $6l$ exponential operations as compared to $2(n-l)$ and $(n-l)$ exponential operations of IKA.1 and IKA.2 protocol, where l is number of leaving members. However, in case of number of messages, our *leave-partition* protocol is inferior to IKA.1 and IKA.2 protocols. In the proposed setup protocol, the total cost of computations has been reduced considerably. For authentication, each group member generates two signatures and performs $2n$ signature verifications, which creates additional cost for authentication. Other existing protocols, for

example, INGM [46], BD [71], Hypercube [12], Octopus [12], Clique (IKA.1 and IKA.2) [77], TGDH [113,115], NAGKA [4], and STR [114], do not provide authentication, our protocol suite provides member authentication at a nominal cost.

Table 5.1: Comparison of group key agreement protocols (IKA.1, IKA.2 and Proposed)

Key Agreement Protocol / Supplementary group Key Agreement Protocol	Efficiency Parameter	IKA.1	IKA.2	Proposed Protocol
Setup	No. of Rounds	n	$n+1$	2
	No. of Messages	n	$2n-1$	$2n$
	No. of exp. Operations	$(n/2)(n+3)-1$	$5n-6$	$3n$
Join	No. of Rounds	2	4	2
	No. of Messages	2	$n+3$	$2(j+3)$
	No. of exp. Operations	$n+1$	$n+3$	$3(j+3)$
Merge	No. of Rounds	$j+1$	$j+3$	2
	No. of Messages	$j+1$	$n+2j+1$	$2(j+3)$
	No. of exp. Operations	$\frac{1}{2}(j^2+2nj+j)$	$n+2j+1$	$3(j+3)$
Leave	No. of Rounds	1	1	2
	No. of Messages	1	1	$4l$
	No. of exp. Operations	$2n-1$	$n-1$	$6l$
Partition	No. of Rounds	1	1	2
	No. of Messages	1	1	$4l$
	No. of Exp. Operations	$2n-l$	$n-l$	$6l$

In proposed setup protocol, each group member performs at most 3 exponential operations, 4 one-way hash function operations, and n XOR operations. Since the operation

dependent on the number of group members is the XOR operation, this way the total cost of computation has been reduced considerably as compared to INGM [46], BD [71], Hypercube [12], Octopus [12], Clique (IKA.1 and IKA.2) [77], TGDH [113,115], NAGKA [4], and STR [114] protocols.

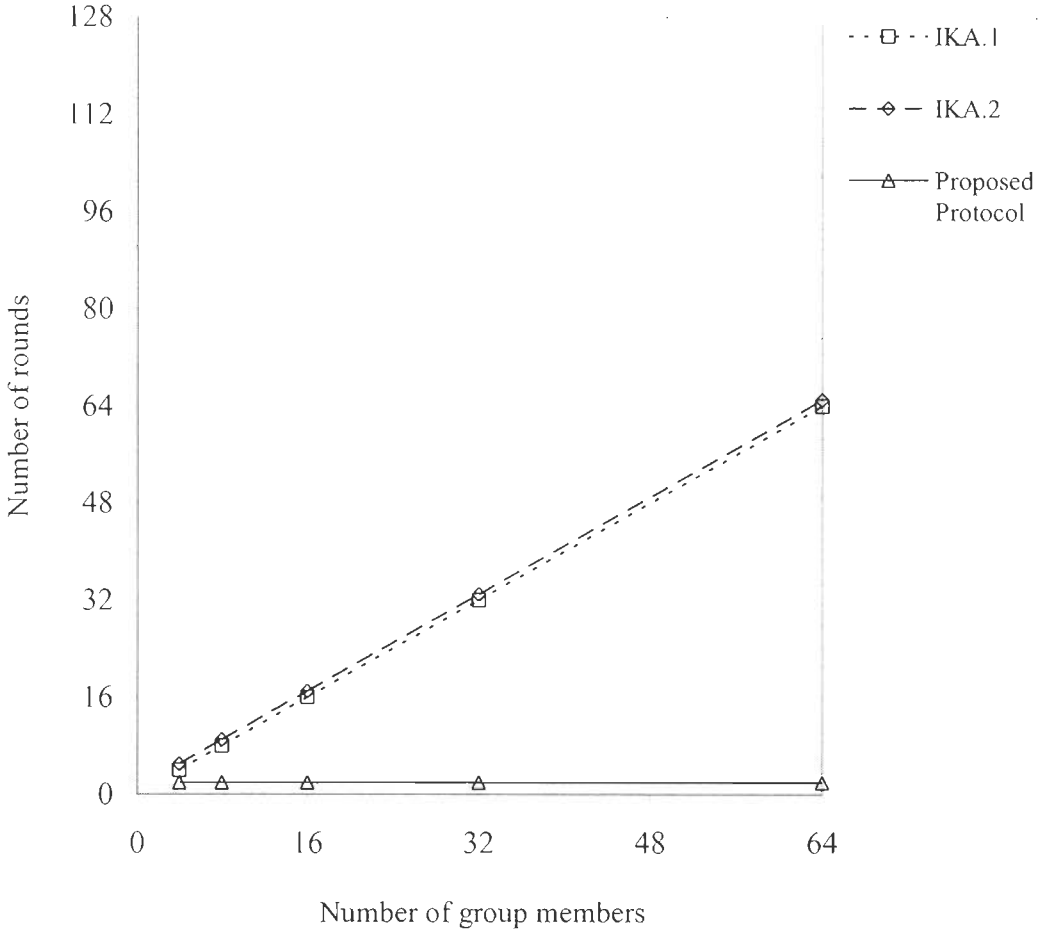


Figure 5.4 Number of rounds Vs. Number of members

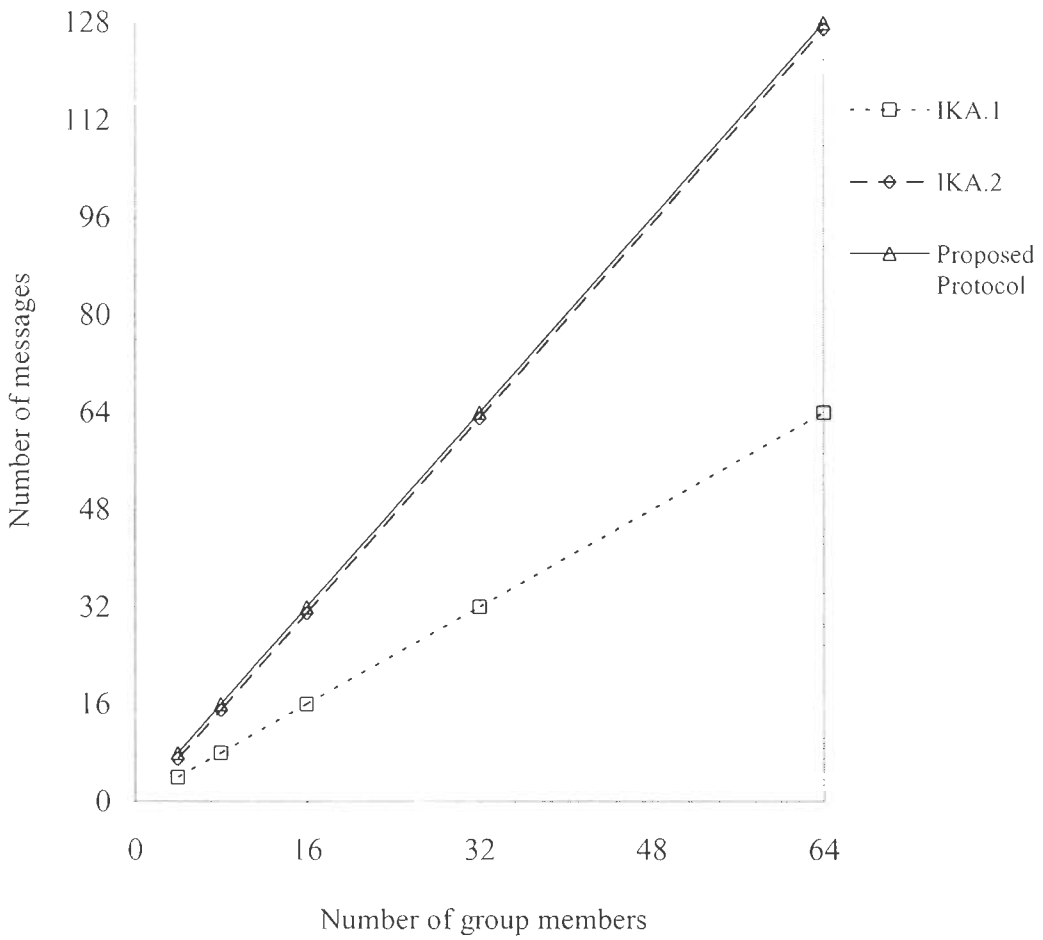


Figure 5.5 Number of messages Vs. Number of group members

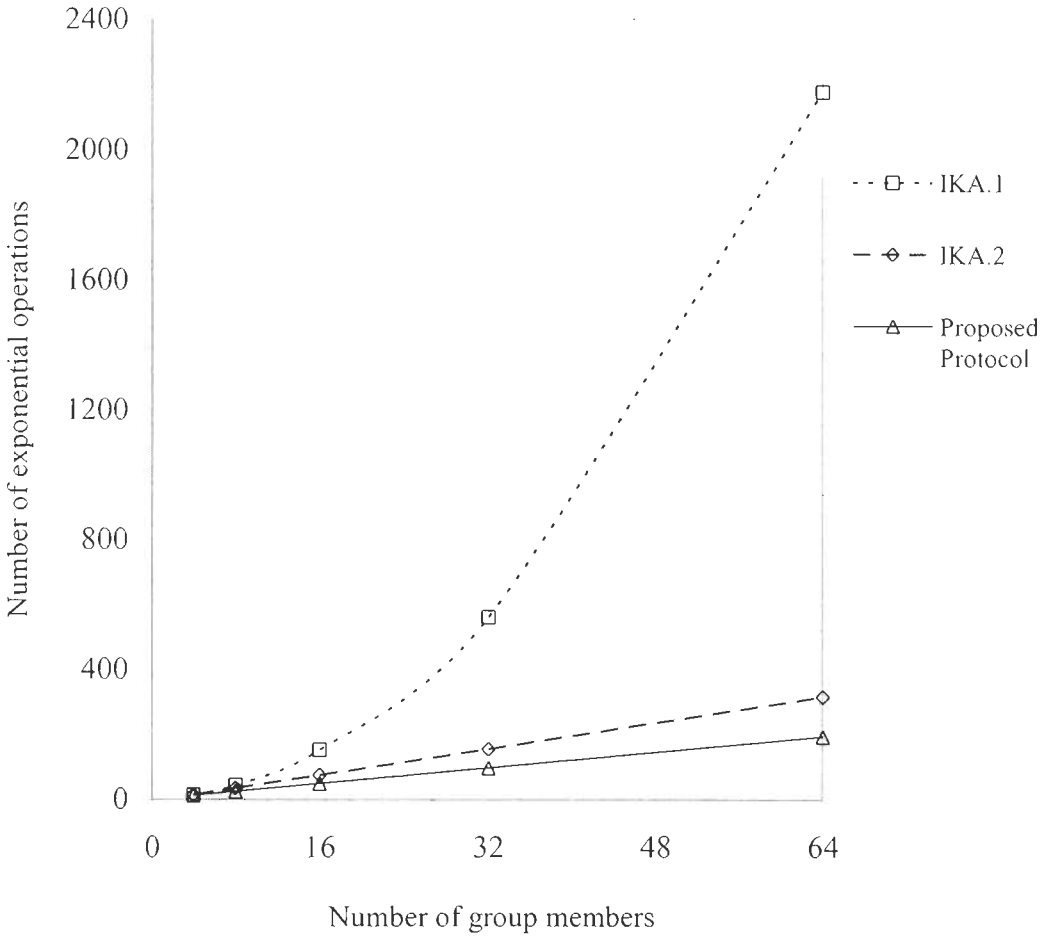


Figure 5.6 Number of exponential operations Vs. Number of group members

5.6 Conclusion

The proposed protocol not only provides efficient algorithms for setup, join (merge) and leave (partition), but also member authentication service. The proposed protocol also has provisions for all valid members to detect errors in communicated messages and stop execution of the protocol immediately as they encounter invalid message from the members who have awry intentions. This helps in eliminating the man-in-the-middle attack in addition to message corruption due to system faults using message verification.

In the proposed protocol suite the members are arranged in a logical ring. The setup algorithm takes initial group as an input and outputs a group key after second round. At the end of each round members are authenticated, and after second round messages are also verified for corruption. If message validity checks fail the protocol terminates immediately, otherwise a group key can be generated and saved by each member in addition to three other parameters, which are used during the dynamic events, i.e., join, leave, multiple join and multiple leave. After computing the group key and three parameters other ephemeral data is erased by each member.

The proposed protocol suite also consists of an algorithm, which can be used for dynamic events join and merge. The algorithm takes the initial group and joining member(s) as an input and outputs the group key and three other parameters for each member. The algorithm also takes two rounds. After each rounds members are authenticated, and after second round messages are also verified for corruption. If message validity checks fail the protocol terminate immediately, otherwise a group key can be generated and saved by each member in addition to other three parameters, which are used during the future dynamic events. After computing the group key and three parameters other ephemeral data is erased by each member.

The proposed protocol suite also consists of an algorithm for leave and partition events. The leave-partition algorithm also takes two rounds. After each round members are authenticated, and after second round messages are also verified for its corruption. If message validity checks fail the protocol terminate immediately, otherwise a group key can be generated. After computing the group key and other parameters, ephemeral data is erased by each member.

Security analysis of the proposed protocol has been done in random oracle model [67]. For which a number of oracle queries are used, which can be replaced by actual function for practical purpose. Secure signature scheme ($\sum$) and hash function (H) in our protocol may be replaced by ElGamal digital signature algorithm [106] and secure hash algorithm (SHA-1) [24] respectively. The selected pseudorandom number in ElGamal digital signature algorithm plays an important role in eliminating the impersonation attack when a secret key is somehow compromised.

In view of the above security and complexity analysis of proposed protocol suite and various existing group key agreement protocols, it is logically concluded that the proposed protocol suite outperforms INGM [46], BD [71], Hypercube [12], Octopus [12], TGDH [113,115], NAGKA [4], STR [114] as well as Clique protocols (IKA.1 and IKA.2) [77] in terms of number of rounds, number of messages, number of exponential operations and round synchronization.

Chapter 6

Conclusions and Scope for Future Work

6.1 Conclusions

With the recent advances in mobile ad hoc network technology, it has been observed that many group-oriented network applications can be easily conducted in this new network environment. For example, in a conference room or in battlefield, users can form an ad-hoc network instantly with their wireless devices, e.g. notebook computers, PDAs, or even cell phones, without requiring any pre-installed cables or base stations. They can use this fast setup ad-hoc network for conducting a videoconference, sharing files or even playing interactive games. The general goal of a secure communication among members is to establish a common secret key (also referred to as a group key), for confidential communication. Usually, a secret group key is established by a group key establishment protocol. In this thesis a secure and efficient group key agreement protocol for mobile ad hoc network have been proposed. The major contributions of our work can be summarized as follows:

1. An extensive investigation of group key establishment protocols has been done. It is observed that only group key agreement protocols can provide a secure and efficient solution for group key establishment in mobile ad hoc networks.
2. Complexity analysis of group key agreement protocols based on two party Diffie-Hellman protocol has been done wherein parameters such as number of rounds, number of messages exchanged, number of exponential operations and round synchronization are taken for comparison purpose. It is concluded that Clique protocols provide the best results among the considered protocols.
3. Complexity analysis of group key agreement protocols based on two party Diffie_Hellman, for dynamic membership events such as join, leave, multiple join,

and multiple leave, has been done. It is concluded that the Clique protocols provide the best results among the considered protocols.

4. An improved secure and efficient group key agreement protocol has been proposed, which not only provides secure and efficient group key establishment but also provides member authentication. It has provisions for all valid members to detect errors and stop execution of the protocol immediately as they encounter invalid message from corrupted members. Further, the proposed protocol has been compared with the Clique protocols. The comparison shows that the proposed protocol is better solution than the Clique protocols for MANET in terms of number of rounds, number of messages exchanged, number of exponential operations and round synchronization.

6.2 Scope for Future Research

The work in this thesis opens up a number of avenues for further work. A number of research issues need to be addressed. Some of these are as follows:

1. Next step to this work may be the design and analysis of 1-round group key agreement protocol with authentication provision for mobile ad hoc network.
2. Node mobility may affect the success rate of group key agreement protocol due to topology changes. It may also affect the data at MAC layer and execution time of protocols, therefore, protocols may be simulated in different nodes' mobility scenarios to observe its effect on data at MAC layer and execution time.
3. Node mobility may severely degrade the performance of the TCP protocol in mobile ad hoc network (MANET). This is due to the inability of the TCP protocol to manage efficiently the effects of mobility. Node movements may cause route failures and route changes and, hence, packet losses and delayed ACKs. The TCP interprets these events as congestion signals and activates the congestion control mechanism. This may lead to unnecessary retransmissions and throughput degradation. In addition mobility may exacerbate the unfairness between competitive TCP sessions. Therefore, investigation may be done for improving the performance under these conditions.
4. The interaction of some features of the 802.11 MAC protocol (hidden-/exposed-station problem, exponential backoff scheme, etc.) with the TCP protocol mechanism (mainly, the congestion control mechanism) may lead to several unexpected and serious problems, which may ultimately affect the performance of group key

agreement protocol. Therefore, the effects of the interaction between MAC protocol and TCP mechanisms may be explored.

5. The TCP congestion window size may have a significant impact on performance of both routing as well as group key agreement protocols. For a given network topology and traffic pattern, there exists an optimal value of TCP congestion window size at which the channel utilization is maximized. However, TCP does not operate around this optimal value and typically grows its average window size much larger, leading to decreased throughput and increased packet losses. This behavior can be explained by considering the origin of packet losses, which in ad hoc networks is completely different from that in traditional wireline network. Therefore, another direction may be to observe the influence of the TCP congestion window size on the performance of group key agreement protocols.

References

[1] A. Ballardie, "Scalable Multicast Key Distribution", RFC 1949, rfc1949.txt, May 1996.

[2] A. Jøsang and G. Sanderud, "Security in Mobile Communications: Challenges and Opportunities," in proc. of the *Australasian Information Security Workshop (AISW'03)*, Adelaide, Australia, February 2003.

[3] A. Ballardie, "Core Based Trees (CBT version 2) Multicast Routing Protocol Specification," RFC 2189, September 1997.

[4] A. Perrig, "Efficient Collaborative Key Management Protocols for Secure Autonomous Group Communication," *International Workshop on Cryptographic techniques and E-commerce*, Hong-Kong, July 5-8, 1999.

[5] A. Perrig, D. Song, and J.D. Tygar, "ELK: A New Protocol for Efficient Large-group Key Distribution," *IEEE Security and Priavcy Symposium*, California, USA, pp. 247-262, May 13-16, 2001.

[6] A. R. Harish, Sreekanth Garigala, Bhaskaran Raman, and Phalguni Gupta, "Feasibility Study of Spatial Reuse in an 802.11 Access Network," *XXVIII URSI General Assembly*, New Delhi, India, Oct 2005

[7] A. Yasinsac, V. Thakur, S. Carter and I. Cubukeu, "A Family of Protocols for Group Key Generation in Ad hoc Networks," in the proc. of *IASTED International Conference on Communications and Computer Networks*, pp.183-187, 2002.

[8] B. Briscoe, "MARKS: Multicast Key Management Using Arbitrarily Revealed Key Sequences," in the *1st International Workshop on Networked Group Communication*, Pisa, November 17-20, 1999.

[9] B. DeCleene, L. Dondeti, S. Griffin, T. Hardjono, D. Kiwior, J. Kurose, D. Towsley, S. Vasudevan, and C. Zhang, "Secure Group Communications for Wireless Networks," in *IEEE MILCOM*, Vol. 1, pp. 113-117, June 2001.

[10] B.. Xie, A. Kumar, and D. P. Agarwal, *Wireless Ad Hoc Networking: Personal-Area, Local-Area and their Sensor-Area Networks*, Auerbach Publications, pp. 535-569, 2007.

[11] C.K.Yeo, B.S.Lee, and M.H.Er, "A Survey of Application Level Multicast Techniques," in Computer Communication, Vol. 27, No. 15, pp. 1547-1568, 2004.

[12] C. Becker and U. Wille, "Communication Complexity of Group Key Distribution," in the *5th ACM Conference on Computer and Communications Security*, California, USA, November 3-5, 1998.

[13] C.Boyd and J.M.G.Neito, "Round-optimal Contributory Conference Key Agreement," in the *proc. of PKC 2003, LNCS 2567*, Springer-Verlag, pp. 161-174, 2003.

[14] C. G. Gunther, "An Identity-based Key Exchange Protocol," *in EUROCRYPT'89*, Belgium, April 10-13, 1989.

[15] C. K. Wong, M. Gouda, and S. S. Lam, "Secure Group Communications Using Key Graphs," *IEEE/ACM Transactions on Networking*, Vol. 8, No. 1, pp. 16-30, February 2000.

[16] C.K. Wong and S.S. Lam, "Keystone: A group Key Management Service," *International Conference on Telecommunication*, Maxico, May 20-25, 2000.

[17] C. Cardeiro and D. P. Agarwal, Ad hoc and Sensor Networks: Theory and Applications, *World Scientific Publishing*, ISBN No. 81-256-681-3; 81- 256-682-1 (paper back), Spring 2006.

[18] C. Shields and J.J. Garcia-Luna-Aceves, "KHIP-A Scalable Protocol for Secure Multicast Routing," in *ACM SIGCOMM Computer Communication Review*, Vol. 29, No. 4, pp. 53–64, October 1999.

[19] C. Shields and J.J. Garcia-Luna-Aceves, "The Ordered Core Based Tree Protocol," *IEEE INFOCOM'97*, Kobe, Japan, April 1997.

[20] Chen Huang, Benxiong Huang, and Yijun Mo, Jianhua Ma, "SRPTES: A Secure Routing Protocol Based on Token Escrow Set for Ad hoc Networks," *IEEE CS proc. of Advanced Information Networking and Applications*, Okinawa, March 2008

[21] D.A. McGrew and A.T. Sherman, "Key Establishment in Large Dynamic Groups using One-way Function Trees," *Technical Report* TR-0755, May 1998.

[22] D. Balenson, D. McGrew, and A. Sherman, "Key Management for Large Dynamic Groups: One-Way Function Trees and Amortized Initialization," draft-balenson-groupkeymgmtoft-00.txt, February 1999. Internet-Draft.

[23] D. Denning and G. Sacco, "Timestamps in Key Distribution Protocols," Communications of the ACM, Vol. 24, No. 8, pp. 533-536, August 1981.

[24] D. E. Eastlake and P. Jones, "Secure Hash Algorithm 1," RFC 3174, rfc3174.txt.pdf September 2001

[25] D. Huang and D. Medhi, "A Key-Chain-Based Keying Scheme For Many-to-Many Secure Group Communication," *ACM Transaction on Information and System Security*, Vol. 7, No. 4, pp. 523-552, November 2004.

[26] D. Pointcheval and J. Stern, "Security Arguments for Digital Signatures and Blind Signatures," in *Journal of Cryptology*, Vol.13. No.3, pp.361-396, 2000.

[27] D. Powell, "Group Communication," *CACM*, Vol.39, No.4, pp.50-53, April, 1996.

[28] D. Steer, L.L. Strawczynski, W. Diffie, and M. Weiner, "A Secure Audio Teleconference System," *CRYPTO'88*, CA, USA, 1988.

[29] D. M. Wallner, E. Harder, and R. Agee, "Key Management for Multicast: Issues and Architecture," *National Security Agency*, RFC 2627, June 1999.

[30] E. Bresson and D.Catalano, "Constant Round Authenticated Group Key Agreement via Distibuted Computing," in *proc. of PKC 2004, LNCS 2947*, Springer-Verlag, pp. 115-129, 2004.

[31] E. Bresson, O. Chevassut and D. Pointcheval, "Provably Authenticated Group Diffie-Hellman Key Exchange – The Dynamic case," in the *proc. of Asiacrypt 2001*, LNCS 2248, Springer-Verlag, Australia, pp. 290-309, Dec. 9-13, 2001.

[32] E. Bresson, O. Chevassut, A. Essiari and D. Pointcheval, "Mutual Authentication and Group Key Agreement for Low-power Mobile Devices," *Computer Communication*, Vol.27, No.17, pp.1730-1737, 2004.

[33] E. Bresson, O. Chevassut and D. Pointcheval, "Provably Authenticated Group Diffie-Hellman Key Exchange," in the *proc. of the 8th ACM Conference on Computer and Communication Security*, pp. 255-264, November 5-8, 2001

[34] G.Anastasi, E.Borgia, M.Conti and E. Gregori, "IEEE 802.11 Ad Hoc Networks: Performance Measurements," in *proc. of the workshop on Mobile and Wireless Networks (MWN 2003)*, Providence, Rhode Island, May 19, 2003.

[35] G. Ateniese, M. Steiner, and G. Tsudik, "Authenticated Group Key Agreement and Friends," *in proc. of 5th ACM Conference on Computer and Communications Security*, ACM Press, San Francisco, CA USA, pp. 17–26, November 3-5, 1998.

[36] G. Chaddoud, I. Chrisment, and A. Shaff, "Dynamic Group Communication Security," in the *6th IEEE Symposium on computers and communication*, July 3-5, 2001.

[37] G. C. Roman, Q. Huang, and A Hazemi, "Consistant Group Membership in Ad hoc Networks," in *proc. of ICSE*, Ontario, Canada, May 12-19, 2001.

[38] G. H. Chiou and W. T. Chen, "Secure Broadcast using Secure Lock," *IEEE Transactions on Software Engineering*, Vol.15, No.8, pp.929– 934, August 1989.

[39] H. Harney and C. Muckenhirn, "Group Key Management Protocol (GKMP) Architecture," RFC 2093, July 1997.

[40] H. Harney and C. Muckenhirn, "Group Key Management Protocol (GKMP) Specification," RFC 2094, July 1997.

[41] H.H. Chu, L. Qiao, and K. Nahrstedt, "A Secure Multicast Protocol with Copyright Protection," *ACM SIGCOMM Computer Communications Review*, Vol. 32, No.2, pp. 42-60, April 2002.

[42] H. Yang, H. Luo, F. Ye, S. Lu and L. Zhang, "Security in Mobile Ad Hoc Networks – Challenges and Solutions," *IEEE Transactions on Wireless Communications*, Feb. 2004.

[43] H. Deng and D. P. Agarwal, *Handbook of Algorithms for Wireless Netowking and Mobile Computing*, Chapman & Hall / CRC, Taylor & Francis Group pp. 937-957, 2006.

[44] Internet Engineering Task Force (IETF) Mobile Ad Hoc Networks (MANET) Working Group Charter, http://www.ietf.org/html.charts/manet-charter.html

[45] IEEE Std. 802.11. Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specification, 1999 edition.

[46] I. Ingemarson, D. Tang, and C. Wong, "A Conference Key Distribution System," *IEEE Transactions on Information Theory*, Vol.28, No.5, pp.714–720, September 1982.

[47] J-P Hubaux, L. Buttyan and S. Capkun, " The Quest for Security in Mobile Ad hoc Networks," in the *proc. of MobiHoc*, Lausanne, Switzerland, June 9-11, 2002.

[48] J. Xie, A. Das, S. Nandi and A. Gupta, "Improving the Reliability of IEEE 802.11 Broadcast Scheme for Multicasting in Mobile Ad hoc Networks," in *IEEE proceedings - communications*, Vol. 153, No. 2, pp. 207 -212, April 2006

[49] J. Katz and M. Yung, "Scalable Protocols for Authenticated Group Key Exchange," in *Advances of Cryptology – Crypto'03*, LNCS 2729, Springer-Verlag, pp.110-125, 2003.

[50] J. Nam, S.Kim, S. Kim and D. Won, "Provably-secure and Communication-Efficient Scheme for Dynamic Group Key Exchange," Available at http://eprint.iacr.org/2004/115.

[51] J. Yao and G. Zeng, "Key Agreement and Identity Authentication Protocols for Ad Hoc Networks," in *proc. of International Conference on Information Technology: Coding and Computing (ITCC'04)*, Nevada, USA, April 5-7, 2004.

[52] J. Zhang, V. Vardharajan and Yi Mu, "A Scalable Multi-service Group Key Management Scheme," in the *proc. of the Advanced International Conference on Telecommunications and International Conference on Internet and Web Applications and Services* (AICT/ICIW 2006), IEEE Computer Society, pp. 172-177, 19-25 Feb 2006.

[53] K. Almeroth and M. Ammar, "Collecting and Modelling the Join/Leave Behaviour of Multicast Group Members in the Mbone," *Symposium on High Performance Distributed Computing*, Syracuse, NY, USA, pp. 209-216, August 6-9,1996.

[54] K. Hoeper and G. Gong, "Identity-Based Key Exchange Protocols for Ad Hoc Networks," in proc. of *Canadian Workshop on Information Theory* (CWIT '05)}, Montreal, Canada, Jun 5-8, 2005

[55] K.H.Rhee, Young-Ho Park, and G. Tsudik, "A Group Key Management Architecture for Mobile Ad Hoc Wireless Networks," *Journal of Information Science and Engineering*, Vol.21, pp. 415-428, 2005.

[56] L. Briesemeister and G. Hommel, " Localized Group Membership Service for Ad hoc Networks," in *proc. of International Workshop on Ad hoc Networking*, 2002.

[57] L.E.Moser, P.M.Melliar-Smith, D.A.Agarwal, R.K.Budhia and C.A. Lingley-Papadopoulos, "Totem: A Fault Tolerant Multicast Group Communication System," *CACM*, Vol.39, No.4, pp.54-63, April 1996.

[58] L. Gong and N. Shacham, "Trade-offs in Routing Private Multicast Traffic," *GLOBECOM'95*, November 1995.

[59] L.Dondeti, S. Mukherjee, and A. Samal, "A Distributed Group Key Management Scheme for Secure Many-to-Many Communication," *Tech. Rep. PINTL-TR-207-99*, Department of Computer Science, University of Maryland.

[60] L. R. Dondeti, S. Mukherjee, and A. Samal, "Scalable Secure One-to-Many Group Communication Using Dual Encryption," *Computer Communications*, Vol.23, No.17, pp.1681–1701, November 2000.

[61] L.R. Dondeti, S. Mukherjee, and A. Samal, "Comparison of Hierarchical Key Distribution Schemes," *IEEE Globcom Global Internet Symposium*, 1999.

[62] L.R. Dondeti, S. Mukherjee, and A. Samal, "Survey and Comparison of Secure Group Communication Protocols," Technical Report, 1999.

[63] L. Zhou, Z. J. Hass, "Securing Ad Hoc Networks," *IEEE Transaction on Network*, Special issue on network security, Nov./Dec. 1999.

[64] L. Guang, Chadi Assi, "MAC Misbehavior in the Ad hoc Network," in 18th Canadian Conference on Electrical and Computer Engineering, IEEE CCECE05, Saskatchewan, Canada, May 1-4, 2005.

[65] M.Bellare and P.Rogaway, " Entity Authentication and Key Distribution," in the *proc. of Crypto 1993*, LNCS 773, Springer-Verlag, pp. 231-249, 1993.

[66] M.Bellare and P.Rogaway, " Provably Secure Session Key Distribution: The Three-party Case," in *proc. of STOC 1995*, ACM Press, Las Vegas, USA, pp. 57-66, May 29- June 1, 1995.

[67] M.Bellare and P. Rogaway, "Random Oracles are Practical: A Paradigm for Designing Efficient Protocols," in *proc. of ACM CCS'93*, Virginia, USA, Nov. 3-5, 1993.

[68] M.Bellare, R. Canetti and H. Krawczyk, "A Modular Approach to the Design and Analysis of Authentication and Key Exchange Protocols," in the *proc. of 30th Annual Symposium on the Theory of Computing*, ACM Press, pp. 419-428, 1998.

[69] M. Blum and S. Micali, "How to Generate Cryptographically Strong Sequences of Pseudo-Random Bits," *SIAM Journal of Computers*, Vol.13, pp.850-864, 1984.

[70] M. Boulkenafed, D. Sacchetti, and V. Issarny, "Using Group Management to Tame Mobile Ad hoc Networks," in *proc. of the IFIP TC8, Working Conf. on Mobile Info. Systems*, Oslo, Norway, Sep. 15-17, 2004.

[71] M. Burmester and Y. Desmedt, "A Secure and Efficient Conference Key Distribution System," *EUROCRYP'94*, LNCS(950), Perugia, Italy, pp.275–286, May 9-12, 1994.

[72] M. Mishra and A. Sahoo, "An 802.11 based MAC Protocol for Providing QoS to Real Time Applications," *IEEE International Conference on Information Technology (ICIT'07)*, India, Dec 17-20, 2007.

[73] M. Moharrum, R. Mukkamala and M. Eltoweissy, "CKDS: An Efficient Combinatorial Key Distribution Scheme for Wireless Ad Hoc Networks," in the *proc. of IEEE International Conference on Performance, Computing and Communication*, pp.631-636, 2004.

[74] M.J. Moyer, J.R. Rao, P. Rohatgi, "Maintaining Balanced Key Trees for Secure Multicast," Internet Draft, June 1999.

[75] M. K. Reiter, "Distributing Trust with Rampart Toolkit," *CACM*, Vol.39, No.4, pp.71-74, April 1996.

[76] M. Steiner, G. Tsudik, and M. Waidner, "Diffie-Hellman Key Distribution Extended to Group Communication," *3rd ACM Conference on Computer and Communications Security*, Delhi, India, pp. 31–37, March 14-16, 1996.

[77] M. Steiner, G. Tsudik, and M. Waidner, "CLIQUES: A New Approach to Group Key Agreement," *in proc. of 18th International Conference on Distributed Computing Systems (ICDCS'98)*, IEEE Computer Society Press, pp. 380–387, Amsterdam, Netherlands, May 1998.

[78] M. Waldvogel, G. Caronni, D. Sun, N. Weiler, and B. Plattner, "The VersaKey Framework: Versatile Group Key Management," *IEEE Journal on Selected Areas in Communications (Special Issues on Middleware)*, Vol.17, No.9, pp.1614–1631, September 1999.

[79] N. Banerjee and S. K. Das, "Multicasting in UMTS: Effect of Mobility on Tree Maintenance," in proc. of the *5th IFIP International Conference on Mobile and Wireless Communications Networks* (MWCN), Singapore, Oct 2003.

[80] N. Asokan and P. Ginzboorg, "Key-agreement in Ad-hoc Networks," *Journal of Computer Communication*, Vol. 23, No. 17, pp.1627-1637, Nov. 2000.

[81] P. Bhagwat, B. Raman, and D. Sanghi, "Turning 802.11 Inside-Out", *2nd Workshop on Hot Topics in Networks (HotNets-II)*, Cambridge, MA, USA, 20-21 Nov 2003.

[82] P. Sethi and G. Barua, "CRESQ: Proving QoS and Security in Ad hoc Networks," PDP 2003, Italy, Feb 2003.

[83] R. Bhasker, "Group Key Agreement in Ad Hoc Networks," *Technical Report 4832*, INRIA, Rocquecourt, France, 2003.

[84] R. Canetti, J. Garay, G. Itkis, D. Micciancio, M. Naor, and B. Pinkas, "Multicast Security: A Taxonomy and Efficient Constructions," *IEEE INFOCOM*, pp. 708–716, March 1999.

[85] R. Molva and A. Pannetrat, "Scalable Multicast Security in Dynamic Groups," in the *6th ACM Conference on Computer and Communication Security*, November 1999.

[86] R. Mukherjee and J.W. Atwood, "SIM-KM: Scalable Infrastructure for Multicast Key Management," *IEEE Local Computer Networks - LCN'04*, pp 335–342, November 2004.

[87] R. Mukherjee and J.W. Atwood, "Proxy Encryptions for Secure Multicast Key Management," *IEEE Local Computer Networks - LCN'03*, October 2003.

[88] R.Mukherjee and J.W.Atwood, "Scalable Solutions for Secure Group Communications," in *International Journal of Computer and Telecommunications Networking*, Elsevier North-Holland, Inc., NY, USA, Vol. 51, No. 12, pp. 3525-3548, August 2007.

[89] R. Mukherjee, "Secure Group Communication," *Doctoral Thesis*, Concordia University, Montreal, Canada, January 2005.

[90] R. Oppliger and A. Albanese, "Distributed Registration and Key Distribution (DiRK)," in the *proc. of the 12th International Conference on Information Security* (IFIP SEC'96), 1996.

[91] R. Poovendram, S. Ahmed, S. Corson, and J. Baras, "A Scalable Extension of Group Key Management Protocol," *2nd Annual ATRIP Conference*, pp. 187–191, February 1998.

[92] R. Rivest., "The MD5 Message-Digest Algorithm," April 1992. RFC 1321.

[93] R. van Renesse, K. P. Birman and S. Mafais, "Horus: A Flexible Group Communication System," *CACM,* Vol.39, No.4, pp.76-83, April 1996.

[94] S. Deering, "Host Extensions for IP Multicast," *RFC 1112*, 1989.

[95] S. Maki, T. Aura, and M. Hietalahti, "Robust Membership Management for Ad hoc Groups," in *proc. of Nordic Workshop on Secure IT Systems*, 2000.

[96] S. Mittra, "Iolus: A Framework for Scalable Secure Multicasting," *ACM SIGCOMM*, 1997.

[97] S. P. Mohanty, R. Sheth, A. Pinto, and M. Chandy, "CryptMark: A Novel Secure Invisible Watermarking Technique for Color Images", in *proc. of the 11th IEEE International Symposium on Consumer Electronics (ISCE)*, pp. 1-6, 2007.

[98] S. Rafaeli and D. Hutchison, "Hydra: A Decentralized Group Key Management," in the *11th IEEE International WETICE: Enterprise Security Workshop*, June 2002.

[99] S. Setia, S. Koussih, S. Jajodia, and E. Harder, "Kronos: A Scalable Group Re-keying Approach for Secure Multicast," *IEEE Symposium on Security and Privacy*, May 2000.

[100] S.Xu and T.Saadawi, "Does the IEEE 802.11 MAC Protocol Work Well in Multihop Wireless Ad Hoc Networks?" *IEEE Communication Magazine*, Vol.39, No.6, pp.130-137, June 2001.

[101] S.Xu and T.Saadawi, "Revealing the Problems with IEEE 802.11 MAC Protocol in Multihop Wireless Networks," *Computer Networks*, Vol.38, No.4, March 2002.

[102] S. Yeo, Rajkumar Buyya, Marcos Dias de Assunção, Jia Yu, Anthony Sulistio, Srikumar Venugopal, and Martin Placek, The Handbook of Computer Networks, ISBN: 978-0-471-78461-6, John Wiley & Sons, New York, USA, 2007.

[103] S. Zhu, S. Setia, S.Xu and S. Jajodia, "GKMPAN: An Efficient Group Rekeying Scheme for Secure Multicast in Ad hoc Networks," in *proc. of International*

Conference on Mobile and Ubiquitous Systems: Networking and Services, pp.42-51, 2004.

[104] T. Ballardie, I.P. Francis, and J. Crowcroft, "Core Based Trees: an Architecture for Scalable Inter-domain Multicast Routing," *ACM SIGCOMM*, pp. 85–95, 1993.

[105] T. Dunigan and C. Cao, "Group Key Management," *Technical Report* ORNL/TM-13470, 1998.

[106] T. ElGamal, "A Public-Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms," *IEEE Transactions on Information Theory*, July 1985.

[107] T. Hardjono, B. Cain, and I. Monga, "Intra-domain Group Key Management for Multicast Security," *IETF Internet draft*, September 2000.

[108] T. Kaya, G. Lin, G. Noubir, and A. Yilmaz, "Secure Multicast Groups on Ad hoc Networks," in *proc. of ACM workshop on security of ad hoc and sensor networks*, pp.94-102, 2003.

[109] V. Shoup, "On Formal Models for Secure Key Exchange," in *Technical Report RZ 3120*, IBM Zurish Research Lab., 1999.

[110] W. Diffie and M.E. Hellman, "New Directions in Cryptography," *IEEE Transactions on Information Theory*, Vol.22, pp.644–654, November 1976.

[111] X. Du, Y. Wang, J. Ge and Y. Wanf, "A Group Key Establishment Scheme for Ad Hoc networks," in *proc. of the 17th International Conference on Advanced Information Networking and Applications (AINA'03)*, pp. 518-520, 2003.

[112] Y. Challal, H. Bettahar, and A. Bouabdallah, "SAKM: A Scalable and Adaptive Key Management Approach for Multicast Communications," *ACM SIGCOMM Computer Communications Review*, Vol.34, No.2, pp.55–70, April 2004.

[113] Y. Kim, A. Perrig, and G. Tsudik, "Tree-based Group Key Agreement," *ACM Transaction on Information and System Security*, Vol.7, No.1, pp. 60-96, Feb. 2004.

[114] Y. Kim, A. Perrig, and G. Tsudik, "Communication-Efficient Group Key Agreement," *IEEE Transaction on Computers*, Vol.53, No.7, pp. 905-921, July 2004.

[115] Y. Kim, A. Perrig, and G. Tsudik, "Simple and Fault-tolerant Key Agreement for Dynamic Collaborative Groups," *7th ACM Conference on Computer and Communications Security*, pp. 235–244, November 1-4, 2000.

[116] Y.R. Yang, X.S. Li, X.B. Zhang, and S.S. Lam, "Reliable Group Rekeying: A Performance Analysis," TR-01-21, June 2001.

Author's Research Publications

International Conferences/Workshops

1. Rakesh Chandra Gangwar and Anil K. Sarje, "Complexity Analysis of Group Key Agreement Protocols for Ad Hoc Networks", in the *proc. of International Conference on Information Technology (ICIT-2006)*, IEEE Computer Society, pp.98-99, Bhubneshwar, India, Dec. 18-21 , 2006.

2. Rakesh Chandra Gangwar and Anil K.Sarje," Secure and Efficient Dynamic Group Key Agreement Protocol for an Ad Hoc Network", in the *proc. of IEEE International Symposium of Ad Hoc and Ubiquitous Computing (ISAHUC'06)*, pp.56-61, Surathkal, India, Dec. 20-23, 2006.

3. Rakesh Chandra Gangwar and Anil K. Sarje, "A Comprehensive Study of Mobility Models for Mobile Ad Hoc Networks", in the *proc. of 3rd International Conference on Mobile, Ubiquitous and Pervasive Computing (ObCom'06)*, Vol.2, pp.1-4, VIT, Vellore, TN, India, Dec. 16-17, 2006.

4. Rakesh Chandra Gangwar and Anil K. Sarje, "Practical and Proven Group Key Agreement Protocols for Mobile Ad Hoc Networks", in the *proc. of International Conference on Advanced Communication Systems (ICACS-2007)*, pp.83-94, GCT, Coimbtore, TN, India, Jan. 10-12, 2007.

5. Rakesh Chandra Gangwar and Anil K. Sarje, "Comparative Analysis of Distributed Group Key Establishment Protocols Based on Subgroup Approach", in the *proc. of IEEE International Conference on Emerging Trends in Engineering and Technology (ICETET-2008)*, pp.823-827, Nagpur, MH, India, July 16-18, 2008.

6. Rakesh Chandra Gangwar and Anil K. Sarje, "An Efficacious Group Key Agreement Protocol for Mobile Ad Hoc Networks", in the *proc. of International Conference on Communication, Convergence and Broadband Networking (ICCBN-2008)*, Indian Institute of Science, Bangalore, Karnataka, India, July 17-19, 2008.