GROUP KEY SCHEMES
FOR SECURITY IN MOBILE AD HOC NETWORKS

by

Depeng Li

Submitted in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

at

Dalhousie University
Halifax, Nova Scotia
April 2010

DALHOUSIE UNIVERSITY

FACULTY OF COMPUTER SCIENCE

The undersigned hereby certify that they have read and recommend to the Faculty of Graduate Studies for acceptance a thesis entitled "GROUP KEY SCHEMES FOR SECURITY IN MOBILE AD HOC NETWORKS" by Depeng Li in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

Dated: April 6, 2010

Supervisor:	Dr. Srinivas Sampalli
External Examiner:	Dr. Guang Gong
Examining Committee:	Dr. Keith Johnson
	Dr. Nur Zincir-Heywood

Table of Contents

List of Tables

List of Figures

Abstract

In dynamic peer group communications, security has been in high demand by many applications in recent years. One of the more popular mechanisms to satisfy these security requirements is the group key scheme in which the group key is to be shared by each group communication participant. However, how to establish and manage the group key efficiently in order to protect such communications imposes new challenges - especially when such schemes are to be deployed on resource-limited networks such as Mobile Ad hoc Networks (MANETs). The basic needs of such network settings require that the group key schemes must demonstrate not only high performance but also fault-tolerance. Furthermore, to encrypt group communication messages efficiently is essential.

Therefore, it is anticipated that the contributions of this thesis will address the development of lightweight and high performance key management protocols for group communications while guaranteeing the same level of security as other approaches. These contributions are listed below:

First, two efficient individual rekey schemes, in which most group members process one-way hash functions and other members perform Diffie-Hellman operations, are proposed to obtain performance efficiency.

Second, a periodic batch rekey scheme is proposed to handle the out-of-sync problem resulting from individual rekeying schemes in cases where there is a high rate of group member requests for joining/leaving.

Third, scalable maximum matching algorithms (M2) are designed to incorporate a tree-based group key generation scheme to forward the partial keys to other group members.

Fourth, a hybrid group key management architecture is proposed as well to combine the advantages of centralized and contributory group key schemes.

Fifth, a Fast Encryption Algorithm for Multimedia (FEA-M) is enhanced to overcome the vulnerabilities of its original solution and its former improved variant.

Performance analyses and experimental results indicate that the proposed approaches reduce computational costs and communication overhead as compared to other popular protocols.

Glossary

ACK	Acknowledge
AG	Anonymous Gossip
AODV	Ad hoc On-Demand Distance Vector Routing
ARQ	Automatic Retransmission reQuest
CA	Certification Authority
CDS	Connected Dominating Set
CPU	Central Processing Unit
DARPA	Defense Advanced Research Projects Agency
DBTDH	Decision Binary Tree Diffie-Hellman problem
DDH	Decision Diffie-Hellman problem
DH	Diffie-Hellman
DoS	Denial of Service
DPG	Dynamic Peer Groups
DST-M2	Depth-first Spanning Tree Maximum Matching
ELK	Efficient Large-group Key
EVS	Extended Virtual Synchrony
FEA-M	Fast Encryption Algorithm for Multimedia
FEC	Forward Error Correction
GPS	Global Positioning System

ID-based	Identification-based
IDA	Information Dispersal Algorithm
IETF	Internet Engineering Task Force
IRTF	Internet Research Task Force
KDC	Key Distribution Center
LAN	Local Area Network
LKH	Logical Key Hierarchy
M2	Maximum Matching
MAC	Media Access Control (layer/address)
MAC	Message Authentication Code (algorithm/protocol)
MANETs	Mobile Ad hoc Networks
MAODV	Multicast Ad hoc On-Demand Distance Vector Routing
MBone	Multicast Backbone
MSEC	Multicast Security
NACK	Negative Acknowledge
OS	Operation System
OSI	Open System Interconnection
PDA	Personal Digital Assistant

PIM	Protocol Independent Multicasting
PIN	Personal Identification Number
PKI	Public Key Infrastructure
RALM	Reliable Adaptive Lightweight Multicast
RDG	Router Driven Gossip
rFEA-M	Robust FEA-M
RLE	Routine Length Encode
RMA	Reliable Multicast Algorithm
RMDP	Reliable Multicast data Distribution Protocol
RTP	Real-Time Protocol
SGC	Secure Group Communication
SM2	Synchronous Maximum Matching
SMuG	Secure Multicast Group
STR	Skinny Tree
TCP	Transmission Control Protocol
TGDH	Tree-based Group Diffie-Hellman
VPN	Virtual Private Networks
VS	View Synchrony (Service)
VS	Vertex Shrinking (Algorithm)
WAN	Wide Area Network

Acknowledgements

I would like to express my sincere thanks to my supervisor Dr. Srinivas Sampalli for his extensive academic support.

I would also like to thank Dr. Keith Johnson, Dr. Nur Zincir-Heywood and Dr. Guang Gong for willing to be a part of the committee and taking time to read my thesis.

Special thanks go to my parent, Xiumei Zhang and Tingyuan Li, my wife, Xiaowen Ge and my son Ang Li for their spiritual supports during my Ph.D study at Dalhousie University.

Chapter 1

Introduction

1.1 Group Communications and Security

The late twentieth and early twenty-first centuries have witnessed the rapid proliferation of group communications which facilitate the exchange of information among a number of group application participants. A wide range of group communication applications such as audio, video, and whiteboard has been deployed [1], [2] and then, they have been extended rapidly to fields such as replicated servers providing database/web/time services, tele/video-conferencing, network layer multicasts, multiparty games, white-boards, and distributed simulations, grid computing and the deployment of Virtual Private Networks (VPNs) [38]. The ease and efficiency of group communication have delivered a new level of productivity for business, organizations and individuals.

In general, group communication applications can be classified broadly into two types: *centralized* and *contributory* [10], [19], [26], [38], [44]. Like typical client-server applications, centralized group applications assume the one-to-many communication pattern in which the group size is very large (e.g. thousands of group members) and in which there are a few senders and many receivers. However, in recent years, a paradigm shift from the traditional client-server model to the peer-to-peer model has changed the landscape of network communications. Organized with the peer-to-peer model, contributory group applications, which assume the many-to-many communication pattern, become more and more prosperous. In such applications every group member has the potential to be both a sender and a receiver. Consequently, any group member can be a message source as well as a recipient. Dynamic Peer Groups (DPGs), one kind of contributory group communication application, turns out to be one of the more popular services in business management, military services, emergency rescue and even entertainment activities during the past decade. Unlike large multicast applications, DPGs are peer-based. Due to the nature of peer-to-peer communication, DPGs are relatively small in group size (10s–100s) [6], [25], [26]. Group members in DPGs can be connected across the Internet, Local Area Networks (LAN), Wide Area Networks (WAN) or Mobile Ad Hoc Networks (MANETs) and they can join or leave the group dynamically. Many network applications involve DPGs. Examples of DPGs include peer-to-peer tele/video-conferencing, peer-to-peer interactive applications and online peer-to-peer multiparty games.

Not only do DPGs introduce efficiency but also increase risk as well. The ease of use of DPGs allows authorized users to access resources from anywhere at anytime and also makes it easier for the unauthorized users and malicious programs to attack the DPGs. Security in group communications is critical for many of these collaborative applications. In fact, group communication messages travel through many more links on the network than unicast data does. Consequently, group communication messages have a greater potential for being attacked. A lack of security obstructs the deployment of group communication applications (DPGs etc.). Without security mechanisms, sensitive information cannot be protected and group application privacy cannot be safeguarded [38].

Several organizations and researchers are focussing on security issues in group communications. The well-known Internet Research Task Force (IRTF) sponsors two work groups, the Secure Multicast Group (SMuG) and the Group Security (GSEC), to investigate and solve security problems for group communications. The Internet Engineering Task Force (IETF) has set up a working team as well, namely, Multicast Security (MSEC), to draft and standardize security technology for group communications in networks, wired or wireless. In addition, research projects such as *Horus and Ensemble* at Cornell University [49], [50], [51], *SecureRing* at the University of California, Santa Barbara [24], *Rampart* at AT&T [46], and *Antigone* at the University of Michigan [31] have focussed on secure group communications.

Protecting DPGs requires a layered defense-in-depth security model. The group key scheme is an important part of this model. This practical and efficient solution deploys a symmetric group key shared by all group application participants. With the support of this shared key (group key), group communication data can be encrypted, which means that, being in possession of the group key, group members can be trusted. Since the group key is so critical, group key schemes are necessary to ensure that only current group members are able to get access to the group key. Group key schemes include group key establishment and management schemes, the methods to generate and update the group key, respectively. Two components comprise group key establishment/management: one is the technique to calculate/agree on the unique group key for every group member, and the other is the method for delivering key material (partial keys) to every group member.

Some group key schemes are designed to make every group member agree on the group key in an efficient, secure and timely way. Following the nature of the group communication model, group key schemes can be classified into two types, *centralized* and *contributory*. In brief, the centralized method relies on the key server and the contributory method needs the collaboration of all group members. For example, in the Network Security Service project at the University of Texas, Austin, a centralized group key scheme is implemented to provide a secure group communication service on the Internet. In this project, a key tree method and a scalable and reliable group key management service, *Keystone*, are maintained by a key server which distributes a group key to other group members via secure channels [64], [68], [72]. An example of the contributory group key scheme is the *Secure Spread* project [3] which is a component of the *Dynamic Coalitions* project funded by the *Defense Advanced Research Projects Agency* (DARPA). *Secure Spread* includes the implementation of a few popular contributory secure group key schemes such as Group Diffie-Hellman (GDH) [6] and Tree-based Group Diffie-Hellman (TGDH) [25] over reliable, robust and ordered network settings. This software guarantees the provision of ordered messages, resilience to any sequence of group membership events and fault tolerance within the group key management service.

1.2 Motivation and Objectives

New breakthroughs in wireless have brought more options than ever to the mobile worker. Meanwhile, the significant enhancement of the processing capability for communication devices (e.g. laptops, wearable computers and high-end Smartphones) enables ubiquitous computing. Therefore, it is not a surprise that deploying DPGs in wireless and mobile environments becomes an attractive choice. In such networks as MANETs, mobile nodes establish routes dynamically among themselves to form their own network on the fly without an existing infrastructure and thus make a good choice for DPGs. However, the wireless and mobile nature of such networks makes it more difficult to secure network communication in MANETs than in classical networks such as WAN and LAN.

Firstly, most mobile networks lack a native infrastructure. Hence, they pose non-trivial challenges for the deployment of group key schemes. Traditional centralized schemes which rely on an on-line key server cannot be a practical choice because of the lack of infrastructure in such networks.

Secondly, such networks have stringent resource constraints. Some low-end mobile nodes tend to be restricted in their computational capability and cannot perform many and frequent computational-intensive operations such as public key cryptographic operations. Furthermore, the communication bandwidth is limited as well. Most previous contributory group key schemes with high computational cost and huge communication overheads cannot be deployed on such networks. We notice as well that even the most efficient contributory group key schemes such as TGDH [25] and STR [26] require too many public key operations which result in heavy-weight computations. Performance consideration in deploying group key schemes in resource-constrained environments drives the cryptographic/security designer to propose more efficient methods for wireless and mobile clients.

Thirdly, such networks generally have an open peer-to-peer architecture, sharing the wireless medium and a highly dynamic network topology. Unlike wired networks where an adversary must gain physical access to the network, it is easier for illegitimate users and malicious adversaries to access the wireless channel. As a result, damage, such as leaking secret information, is more likely to occur.

Therefore, novel group key architectures, algorithms, or protocols should be proposed which can not only handle the absence of a key server but also work efficiently to satisfy stricter performance requirements. The focus of this research is to develop a suite of efficient and reliable group key schemes/architectures for DPGs. Furthermore, strong and efficient encryption algorithms must be studied as well to present confidential services. Consequently, the objective of this research is to provide key management schemes for group communications which can present an access control service for group key and group communication data.

1.3 Contributions and Proposed Work

This thesis is devoted to promoting performance and fault-tolerance for group key schemes. The purpose of this research is to provide a set of group key schemes to safeguard group communication data in a proactive way. Since the one-way hash function (Hash) is computationally more efficient than the Diffie-Hellman operation while providing the same level of security, it is utilized in my proposed solutions. The main contributions are given below.

1. The first contribution provides two efficient individual contributory group key management schemes. The key ideas in the proposed approaches are as follows:

a) To improve the efficiency of TGDH, the proposed approach updates the group key via a Hash function for a joining group member and postpones the Diffie-Hellman-based key updates until group members leave; it is called TGDH-H.

b) To enhance the efficiency of a), the proposed method presents an efficient individual rekey scheme in which most group members process one-way hash functions and only a few members perform Diffie-Hellman operations; it is called TGDH+.

2. The second is a periodical batch rekeying scheme. Proposed group key management schemes such as TGDH-H and TGDH+ result in an out-of-sync problem when the rate of join/leave requests is too high. To deal with this casual scenario, a periodic batch rekey method which updates the group key at the end/start of the periodic interval will be presented. The out-of-sync problem can be solved and the rekey group key scheme should not crash when the rate of join/leave requests is too high since the periodic rekey strategy does not respond to every single group membership change.

3. The third contribution is a scheme to deliver key material for contributory group key establishment schemes. Firstly, for tree-based group key agreements, to deliver partial key messages efficiently, two Maximum Match (M2) algorithms are proposed to accommodate binary key tree construction in MANETs. While most previous algorithms, [4] [62], have been primarily sequential, M2 is a parallel algorithm. Secondly, M2 algorithms require that each member only keeps track of its matched partner instead of remembering the entire group member list which reduces the storage cost for every node.

4. The fourth proposal is a novel hybrid group key management architecture to accommodate the centralized and contributory group key management schemes. In particular, the method enables fast switching between centralized and contributory schemes with minimal communication costs and computational overheads and uses fewer keys.

5. The fifth is concerned with encryption algorithms. In order to encrypt multimedia streams efficiently which can be delivered in real-time group environments, a Fast Encryption Algorithm for Multimedia (FEA-M) has been proposed [70], [69]. Crypt-analysis of this technique, [32], [33], [34], [35], has identified its weaknesses and an improved variant has been suggested, [35]. In this thesis, crypto-analyses are utilized to identify further weaknesses in the original FEA-M as well as in the improved variant. The proposed solution provides message integrity, guarantees zero packet loss and protects against specific known plaintext attacks.

In summary, the research in this thesis proposal has been developed to provide efficient and fault-tolerant group key schemes to secure messages in DPGs.

Future research will focus on the group key schemes in Byzantine mode to manage the group key in case where inside group members are compromised.

1.4 Outline

The remainder of this thesis is divided into six chapters. The security background of group communications is introduced in Chapter 2. Previous group key schemes, FEA-M and cryptographic primitives are demonstrated in Chapter 3. Contributions for the thesis are described in Chapter 4. Performance analyses, experimental results and security discussions are presented in Chapter 5. An integrated solution including all my proposals is provided in Chapter 6. Concluding remarks are summarized in Chapter 7.

Chapter 2

Scope and Background Knowledge

2.1 Group Communication in MANETs

Group-oriented systems have been developed and implemented for a long time. Serveral group communication systems such as *Horus*, *Ensemble* and *Spread* are deployed in classical network settings such as LAN, WAN and the Internet. In order to launch successful group communications, guaranteeing fault-tolerant and replicated services is required, in which the following functionalities are necessary:

- *Group membership service:* all current alive and connected group members are put on the list and this list should be delivered to every current group member. Furthermore, this list should be updated whenever the group member joins or leaves.

- *Reliable and ordered message delivery service.*

MANETs have experienced difficulty satisfying the above requirements due to such characteristics as link failures, network partitions and node crashes. Although not easy to do, recent research and development has introduced successful implementations of a set of group communication systems in MANETs based upon the delivery of realiable multicast and synchronized update group membership lists. *JazzEnsemble* [76], for example, utilizes deterministic approaches, in which every group member is associated with a fuzzy level. The level indicates the group node's ability to process and deliver messages. This method copes with highly mobile nodes and handles wireless communications successfully. Unlike *JazzEmble*, *Random walk* [77] is a different example which explores probabilistic approaches rather than in a defined way. In this method, a group leader is responsible for collecting the entire group membership view and every group member can host an entire or partial group member list. Messages can be delivered around the group settings with a high probabilistic rate. Both of the solutions can provide a reliable and ordered multicast delivery and also the group membership service as well. For details about how to implement them and how they can provide group services, please refer to [76] and [77].

2.2 Secure Group Communication

As a result of participation in collaborative group services in LANs, WANs, the Internet or MANETs, network users are concerned more and more about security. Without a security mechanism to safeguard communication data, group communication messages can be intercepted and fake group messages can be delivered around network settings by attackers at anytime.

Attacks or threats can be launched at any layer of a protocol stack suite, such as the Open System Interconnection Reference Model (OSI) or TCP/IP. In the lower layers of MANETs, security mechanisms should protect communication nodes against threats such as physical tampering, spread-spectrum techniques, frequency hopping, and interleaving. At higher layers, attacks to group communications include passive eavesdropping, active message interception, message insertion, message replay, and denial of service, which compromise Secure Group Communication (SGC) fundamental security properties such as confidentiality, integrity and authentication. This thesis focuses mainly on security mechanisms deployed at the network layer to protect network or higher layers. Providing solutions to secure lower layers is outside the scope of this thesis.

2.2.1 Security for Group Communication

A secure group communication system should provide security services for group communication abstractions. These group activities range from group initialization to group message delivery, group member joining/leaving, group organization, and consistent viewing of group events. Several security mechanisms are provided to safeguard group communication. In general, there are three main categories, as listed below:

Access control to the group

Access control is fundamental for secure group communication; it admits legal clients and blocks illegal or malicious users based upon the group member admission police. Actual research interests include how to admit a new group member and how to make the decision to expel a client. Of interest as well is trust managements in which the possession of the group key verifies that the node is trusted.

Group authentication

The counterpart for the admission of the new group member is how an existing group member convinces outsiders of its specific group member status. A group signature mechanism makes the authentication work. Other proposals such as ID-based authentication can be deployed as well. With the deployment of these methods, impersonation or man-in-the-middle cannot comprise either the group membership protocol or the group view update.

Group Key Agreement/Management schemes

The group key scheme plays a key role in the fundamental infrastructure of SGC to satisfy confidentiality, integrity and authentication. In SGC, eavesdropping compromises confidentiality; malicious actions such as insertion, deletion, and modification of group messages damage integrity; man-in-the-middle attacks damage authentication. With the support of other cryptographic mechanisms such as encryption, Message Authentication Code (MAC), and digital signing, the group key could protect group communication atomic against attacks and satisfy the security goals listed earlier.

In fact, many security solutions (such as secure routing protocols) are designed with the assumption of the pre-deployment of a group key. Therefore, the group key scheme works not only as one of the more important mechanisms, but also plays the cornerstone role in SGC to ensure security for both group member collaborations and group communication data transportation.

The focus of this thesis is to propose group key schemes for group communications so that only current group members can hold the current group key. Specifically, contributions are made to schemes which make the group key accessible only to current group members and hence no other users can decrypt the message. Providing solutions for group authentication or access control to the group is outside the scope of this thesis.

2.2.2 Security Model

Group key schemes can be generated with or without threats of insider attackers. For these two cases, corresponding solutions follow two models: the fortress security model and the model against Byzantine failures (i.e., arbitrary, including malicious). The former refers to the secure mechanism with assumptions that all group members behave correctly and indeed, are trustful. The latter assumes that insider nodes/members can be compromised / fail as well. Both of them demonstrate the realities of GCSs. In general, most cryptographic designs such as Diffie-Hellman (DH), TGDH etc., assume that the node possessing a valid secret key is trusted, which means that insiders are trustful. This is an example of the fortress model. Meanwhile, in some scenarios, nodes in the networks can be tampered/captured by malicious users as well, which means that the inside member may be un-trustful. This would be an example of Byzantine model.

Proposals using the fortress demonstrates efficiency. By contrast, with the Byzantine model, in order to expel compromised inside nodes, solutions which escort group communications safely require heavy communication overhead and demand more network resources. So far, proposals presented for classical networks such as LANs or WANs can achieve the security objective if the compromised members comprise no more than 1/3 of the group size. Otherwise, solutions will fail due to high traffic communication volumes. MANETs characteristics demonstrate a lack of bandwidth. Furthermore, links frequently corrupt, nodes sometimes crash and networks partition. In a word, the resources are limited. So far, no successful solutions have been implemented for MANETs. Therefore, in this thesis, the fortress model is focused upon and research against Byzantine attacks will be studied in the future.

In the fortress security model, it is necessary to consider only outsider adversaries. They can be divided into two types, active and passive. Passive outsider adversaries capture group communication data to launch eavesdropping attacks. Group key schemes can be employed to protect against this attack. Active outsider adversaries use malicious techniques which include injecting, modifying, delaying and deleting group communication messages to break authentication and integrity. A man-in-the-middle attack can be launched as well by outsider adversaries. The corresponding security technique is to sign the messages digitally with the public key of the sender and then the receivers verify the signings [32]. A sequence number or timestamp should be included as well to indicate the freshness of rekey messages. Since these kinds of algorithms are well known and well developed, detailed implementation will not be described in this proposal.

This research cannot be deployed against attacks such as traffic analysis, Denial of Service (DoS), Operation System (OS) attacks, and Byzantine attacks. To overcome them, please refer to the corresponding solutions.

2.2.3 ID-based Authentication Used for Group Member Authorization

Before the introduction of my solutions, trust management between the group participants or that between the existing group member and a new joining group member should be discussed. An ID-based technique is one of the methods who can authorize users which plan to join subsequent group sessions. It means that before a user sends out join requests, it is necessary to apply for the authorization procedure first. After the private key and the public key have been issued successfully, this user can send out a join request. So, in the fortress model, group key management is our objective and authentication can be provided by ID-based DH authentication. In detail, every group member should be issued a public key / private key pair from Certification Authority (CA) server via secure channel (eg. Physical Touch, Master Card, Token, etc.). Notice that the CA server should not have to be available all the time. Afterwards, these nodes are legible to join any group. In order to participating any group, it is simply a matter of sending out the request package including its public ID signed by its RSA public key. One of the current group members (called sponsor) can respond to the request and reply with its partial key to process the group member admission (please refer to section 3.1.4 for details). Therefore, with the issued public/private key pair, the node can associate and then finally participate in the group.

2.2.4 Secure Group Communication (SGC)

Secure Group Communication (SGC) refers to scenarios in which group members can receive and send messages to others in such a way that outsiders are unable to obtain any information even when they are able to intercept the messages [74]. Group members are group application participants who send/receive group messages with other members to fulfill collaborative tasks. Two communication patterns, one-to-many or many-to-many, can be used to send/receive group messages. In the one-to-many type, one group member sends messages and a number of members receive messages. By contrast, in the many-to-many type, each group member can be the message sender and receiver simultaneously. Centralized group communication assumes a one-to-many format and contributory group communication, including DPGs, assumes the many-to-many format.

2.2.5 Group Membership Changes

In group communications, the group member can join or leave the group at any time. This feature of group communications is called the *group dynamic*.

In addition to one group member joining/leaving, the group dynamic also involves another kind of behaviour, namely the *bulk* or *bursty* operation in which multiple group members join and/or leave the group simultaneously [38], [44]. Therefore, the group key management scheme should make the group key accessible only to current group members for the following group membership changes [38], [44] (shown in Figures 2.1 - 2.4).

Member join: To take part in present and future group communications a user *joins*

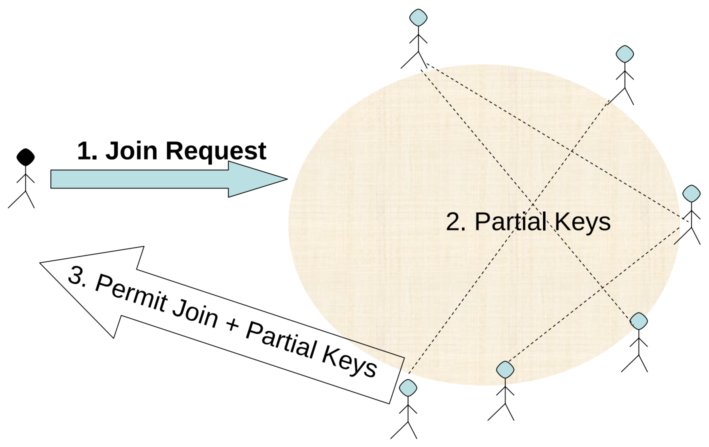

Figure 2.1: Group Membership Changes - Member Join Scenario

the group to become a legitimate group member. The join procedure is processed as mutual authentication between the new node and the node representing the group to admit the new one.

Member leave: During group communications, a group member *leaves* the group. Once a leaving group member sends out the leaving request, a group member, which may be a leader or an average user, will forward its leaving request to all other members. After leaving and before rejoining, the member can no longer obtain (decrypt) group communication messages. If the member is out of service, it is treated as leaving via the heart-beat detection technique.

Group partition: In certain applications, a group may be divided into two or more

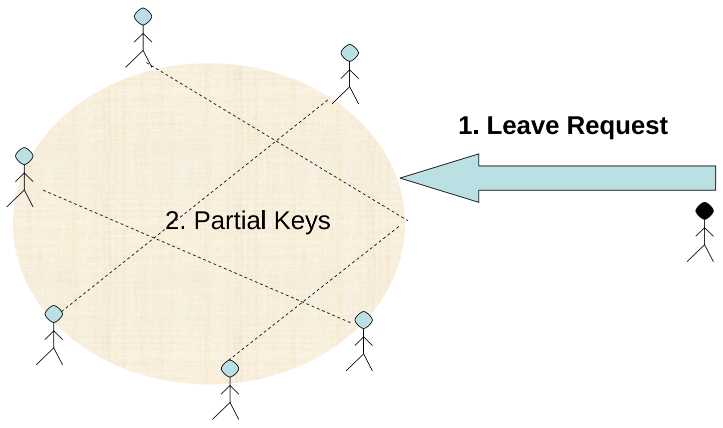

Figure 2.2: Group Membership Changes - Member Leave Scenario

independent subgroups. Another scenario for group splitting is triggered by physical network partitions. The group is then separated into multiple temporary subgroups.

Group merge: Two or more groups may fuse into a larger group in some applications. Moreover, groups previously split due to a network partition may need to merge again because of physical network restorations.

Note that the method for deciding and processing partition/merging operations is determined by certain group applications and is not considered in this proposal. Furthermore, whether a group member leaves the group or not is based upon the group communication semantic which is outside the scope of this thesis. As well, the reasons and mechanisms for expelling group members are outside the scope of this research.

Bursty Operation: Multiple group members join and multiple group members leave the group simultaneously.

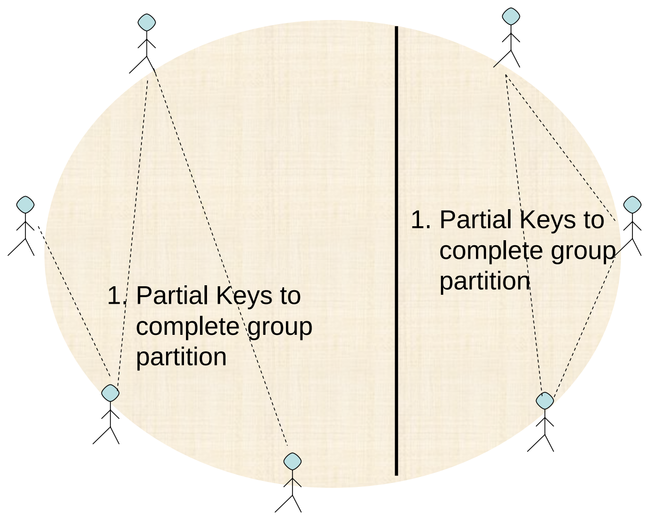
1. Partial Keys to complete group partition
1. Partial Keys to complete group partition

Figure 2.3: Group Membership Changes - Member Partition Scenario

2.2.6 Security Requirements for Group Key Management

In SGC, group members can join/leave at any time and the current group key should be possessed only by current group members. Therefore, it is critical that the future group member and the leaving group member not obtain the current group key. So, when one or more members leave or join the group, the group key should be updated so that just current group members comprehend it. This procedure is called rekeying. Based upon the security properties defined in several systems [10], [38], [44] (shown in Figure 2.5 - 2.9), a list of security requirements for group key management follows.

Group key secrecy/key independence: It must not be computationally feasible for a passive adversary to discover any group key.

Forward secrecy: Previous group members who know contiguous subsets of old group keys must not be able to discover subsequent group keys after they leave the group.

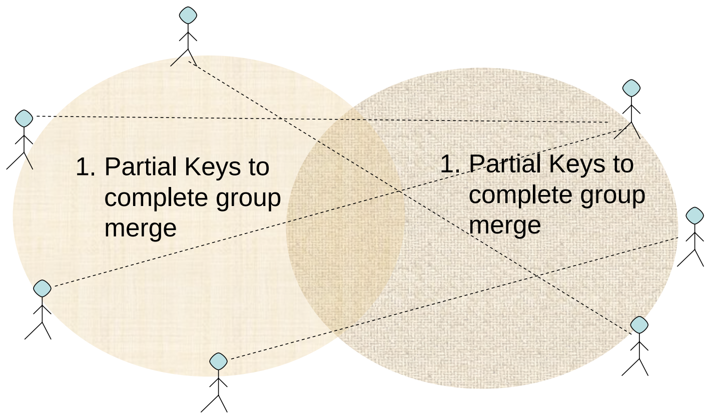
1. Partial Keys to complete group merge
1. Partial Keys to complete group merge

Figure 2.4: Group Membership Changes - Member Merge Scenario

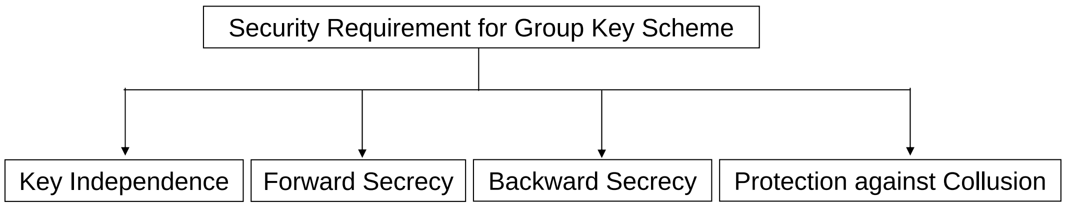
Security Requirement for Group Key Scheme
Key Independence
Forward Secrecy
Backward Secrecy
Protection against Collusion

Figure 2.5: Group Key Security Requirement Summary

Backward secrecy: Current group members who know a contiguous subset of current group keys must not be able to discover preceding group keys.

Protection against collusion: No set of fraudulent users should be able to deduce the current traffic encryption key, which means that nonmembers of the group must not be able to collude and compromise the current group key.

There are two kinds of rekey strategies: *individual rekey* and *periodical batch rekey.*

Individual rekey: the rekey scheme is processed to update the group key for such group membership requests as joining/leaving.

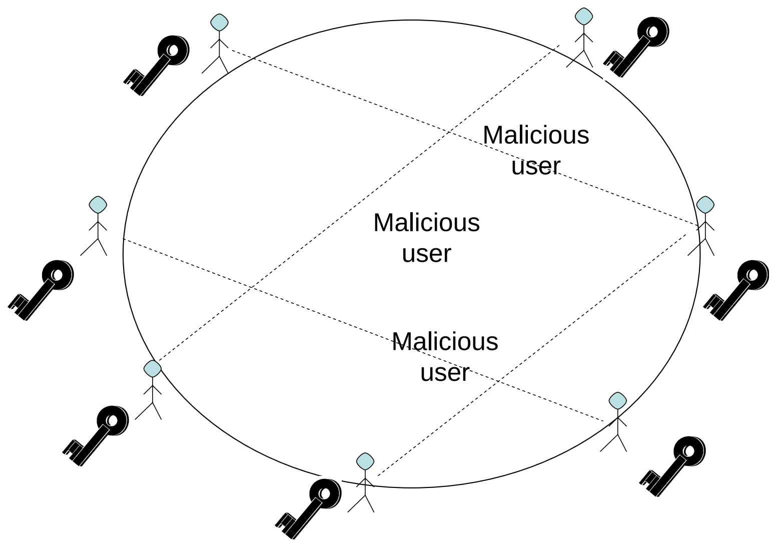

Figure 2.6: Group Key Security Requirement - Key Indepedence

Periodic batch rekey: when a member requests to join or leave the group, rekey operations corresponding to these joining or leaving requests are postponed rather than processed immediately and requests are processed in a batch at the end of each rekey interval so that group members who need to leave can stay longer and new group members have to join later. The purpose is to accumulate more joining and leaving requests so that these multiple requests can be processed in bulk.

The individual rekey strategy introduces inefficiency and out-of-sync problems [68], especially for resource-limited networks. However, it can provide strict backward secrecy and forward secrecy. An individual rekey strategy might be deployed most appropriately for financial/military applications which require strict security.

Periodical batch rekeying leads to a *vulnerability window* [68] which is the period of time starting with the joining or leaving request and ending at the completion of the rekey interval. If the vulnerability window is too long, security can be compromised.

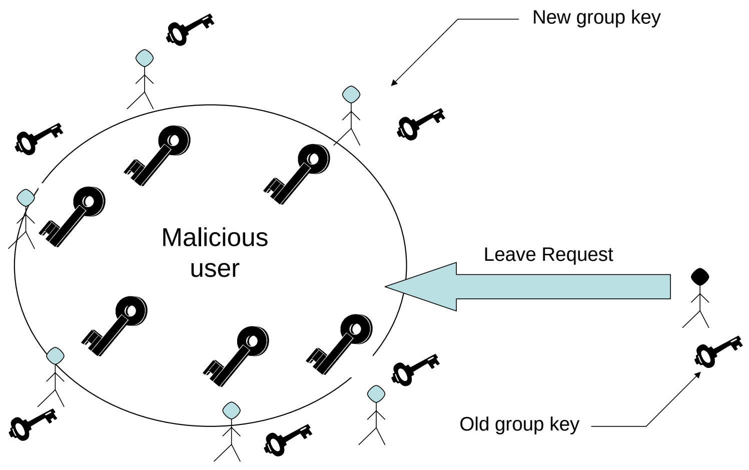

Figure 2.7: Group Key Security Requirement - Forward Secrecy

However, it can alleviate the out-of-sync problem and improve efficiency. Consequently, periodic batch rekeying is a trade-off between the group key security and efficiency. A periodical batch rekeying strategy is used in entertainment/education applications which have security demands which are not so strict.

2.3 Components and Criteria of Group Key Schemes

Figure 2.10 shows three basic functional components of a group key scheme: *registration*, *rekey processing* and *rekey transport*.

A user sends the authorization request to the registration component which could validate/verify the user. The registration component is responsible for processing authorization requests sent from users. In general, a registration component of the group key management scheme can authorize a user employing such techniques as Public

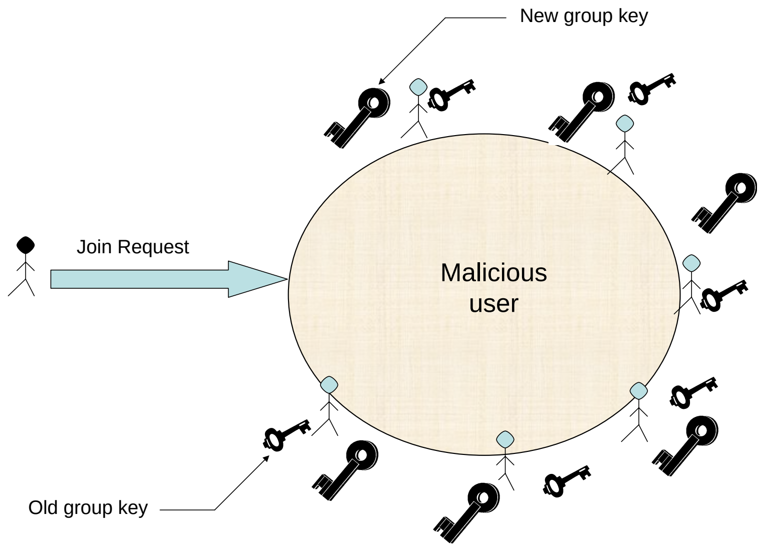

Figure 2.8: Group Key Security Requirement - Backward Secrecy

Key Infrastructure (PKI), ID-based authentication [32], physical touch, master card, token, etc. After the registration component sends secret keys to the authorized user, the user can use these secret keys to obtain the group key. Notice that the registration component is an offline third-party CA. The manner in which the user is authorized is outside the scope of this proposal and might be considered as a subject for future research.

After being authorized, the user is capable of obtaining a group key as long as s/he participates in a group session. The method for delivering the rekey message (partial keys) to the new group member is called rekey transport and the technique for gaining/updating the group key is called rekey processing. The rekey transport component is utilized to deliver rekey messages around the network via reliable and authenticated unicast or multicast services. The rekey processing component is developed to manage/update the group key. This component enables join requests sent

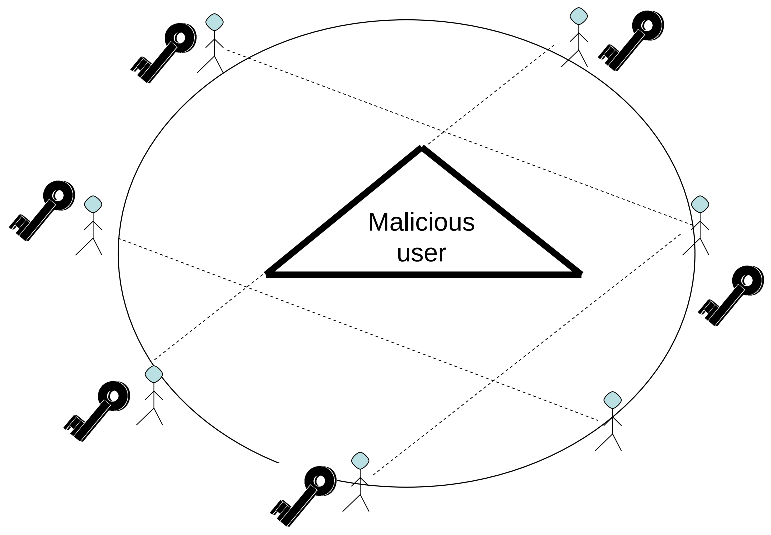

Figure 2.9: Group Key Security Requirement - Against Collusion

from the authorized user and leave requests sent from the group member to be processed. More importantly, by executing the rekey processing scheme (component), every group member can agree on a new identical group key.

As shown in Figure 2.10, generally, the rekey processing component includes two generic components: group key establishment and group key management schemes. The mechanism to generate a group key is called the group key establishment scheme. The contributory group key establishment scheme has been called the group key agreement scheme as well [32]. Centralized group key establishment schemes can be called group key distribution schemes [32]. In contributory group key establishment schemes, distributed algorithms forward partial keys over the network. Other group members calculate the group key based upon these partial keys. The method for updating a group key is called the group key management scheme [32], [44] which is utilized when a member leaves or joins the group.

The rekey processing and rekey transport components are installed on the same network service entity/node. However, they do not have to be installed on the same entity/node as the registration component. In general, the registration component is deployed on the Group Control (GC) which is off-line. The rekey processing and rekey transport components are installed on the Key Server (KS). In cases where large numbers of users are required for authorization and where the registration rate is high, distributed multiple GCs are employed to distribute the workload for the authorizing user requests. This mechanism improves the scalability of registration services. Figure 2.11 and Figure 2.12 delineate registration services for the centralized and contributory group key schemes, respectively.

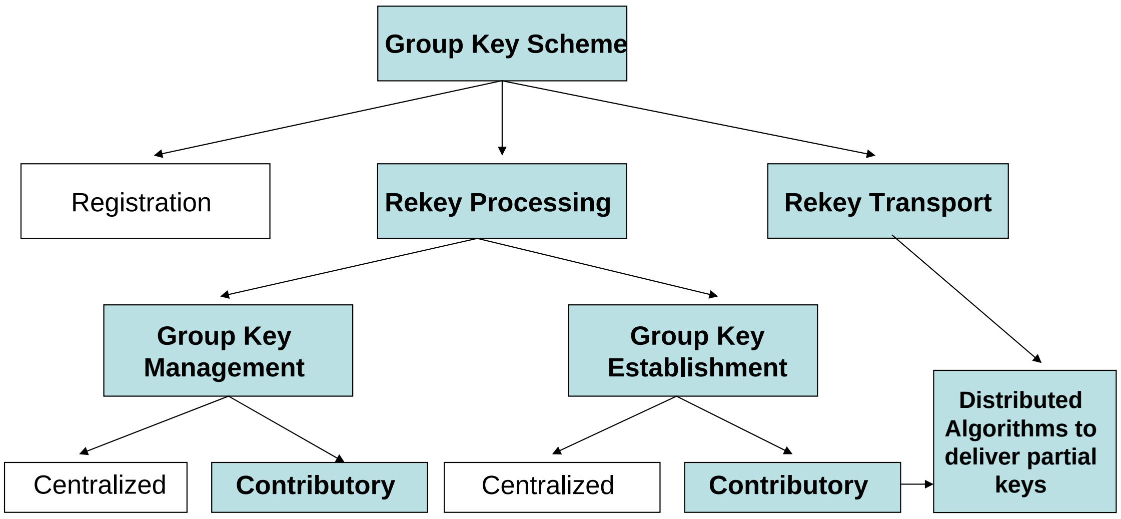

Figure 2.10: Group Key Category

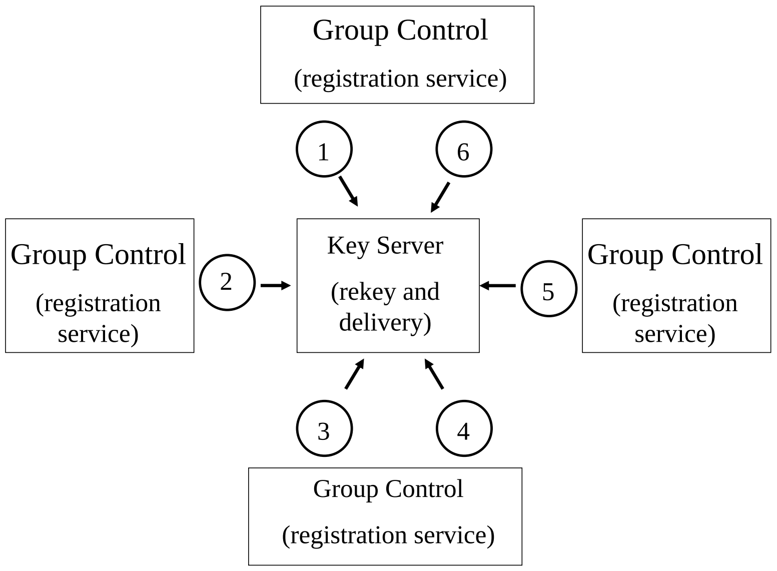

Figure 2.11: Centralized Group Key

2.3.1 Criteria to Evaluate Group Key Management

Efficient group key management schemes should take into consideration not only security but also performance requirements. Previous research [44] has proposed a number of criteria useful for validating and evaluating group key management schemes. In addition to the security property introduced earlier to safeguard group communication messages, the following performance-relevant requirements (shown in Figure 2.13) are provided as well for comparing different group key management schemes.

Service availability/fault tolerance: the fault of a single entity should not prevent the operation of the key management scheme.

Computational cost: to update the group key some cryptographic schemes such as encryption/decryption, Diffie-Hellman key exchange, or one-way hash functions should be used to let all group members generate the unique, new group key. It is desirable that the rekey computation processing cost is low.

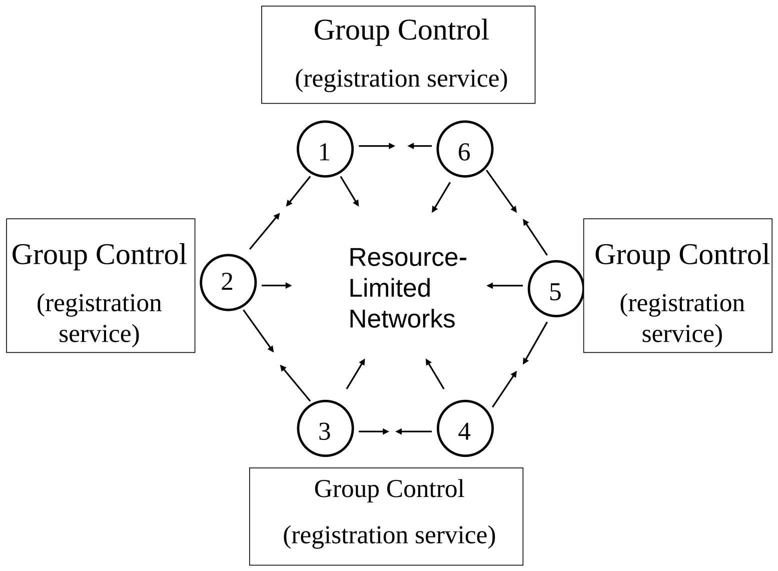

Figure 2.12: Contributory Group Key

Communication overhead: to update the group key, the rekey messages should be delivered around the network so that other group members can use them to update their group keys. It is desirable that the bandwidth utilized for rekey messages be small.

Storage requirement: the number of keys stored by group members and the key server should be small.

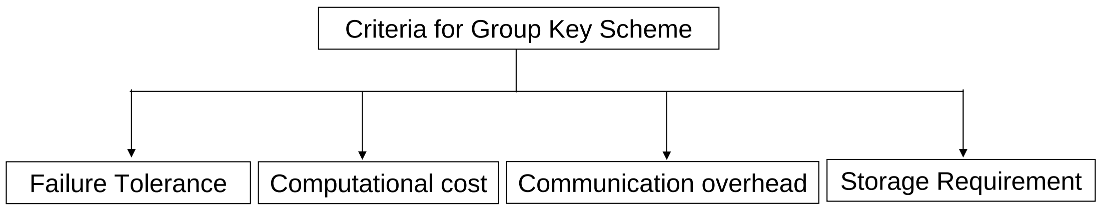

Figure 2.13: Group Key Performance Criteria

2.4 Anticipated Significance of Contributions

This research provides an original theoretical contribution to the study of group key schemes for DPGs and encryption algorithms. The research is concerned with group key management which has been studied for a long while.

Firstly, this research focusses on the performance-relevant enhancements to make Diffie-Hellman-based contributory group key management suitable for resource-limited networks. Issues like fault-tolerance, computational cost, communication overhead and memory consumption are addressed. In particular, the focus of this research is on the efficiency of the contributory group key generation scheme, the contributory group key management scheme and a hybrid architecture, all of which are critical for these group communication applications in resource-limited networks.

Secondly, this research provides insights into practical problems as well including the group member join/leave rate which affects the performance of the group key schemes significantly. The proposed approaches improve the tolerance threshold for the join/leave rate in resource-limited networks by increasing performance by 50 percent.

Thirdly, most experiments designed in this research to test the performance of the proposed approaches are based upon a real data set collected on MBone [1, 2], a popular and practical group communication platform on the Internet enjoyed by various users. The group member behaviour for member join, member leave and the duration time of a member attending the group session is recorded within real time executing environments. The rest of the experiments are achieved via platform Network Simulation 2 (NS2), an extensively used simulation tool.

Chapter 3

Background and Literature Survey

In the following discussion, cryptographic primitives are introduced in section 3.1. Related works for group key establishment and management schemes are described in section 3.2 - 3.4. Previous works related to FEA-M are provided in section 3.5.

3.1 Cryptographic Techniques

3.1.1 Two-Party Diffie-Hellman Key Exchange Scheme

Party A and party B launch the two-party Diffie-Hellman key exchange protocol to generate a common secret key, k, shared by both of them [32]. Assume that p is a large prime and g is a primitive element of *GF(p)*. What's more, the integer pair, $(p,\ g)$ is known by both parties in advance. What follows are the steps required to accomplish the Diffie-Hellman protocol.

Step 1: Party A and B generate random integers, a and b respectively to satisfy $0 < a < p-1$ and $0 < b < p-1$.

Step 2: Party A and B calculate and exchange the result of g^a (mod p) and g^b (mod p) with each other.

Step 3: Party A calculates the secret key $k = (g^b)^a$ (mod p).

party B calculates the secret key $k = (g^a)^b$ (mod p).

Finally, secret key k is shared by party A and party B.

3.1.2 One-Way Hash Function

One-way function $h()$ should satisfy four requirements [28], [32]:

(1) The function $h()$ is known in advance by public users including not only the legitimate participants but also the malicious users.

(2) For an arbitrary length message M, it is computationally efficient to process h(M).

(3) It is not computationally feasible for the malicious user to deduce the input M if s/he can obtain the output of the one-way hash function, h(M).

(4) Function $h()$ provides second pre-image collusion resistance. That is, given a random value x, it is not computationally feasible to find $x' \neq x$ such that $h\ [x] = h\ [x']$.

3.1.3 Hash Chain

A hash chain is a set of sequenced hash values with a linear derivative relationship among multiple one-way hash functions [32].

The hash chain formula is $\mathbf{H}^j(\mathrm{G}) = \mathbf{H}(\mathbf{H}^{j-1}(\mathrm{G}))$ where $\mathbf{H}$ is a one-way hash function and j is an integer. $\mathrm{H}^j(\mathrm{G})$ means using hash function H() j times on a message G. Therefore, if $\mathrm{H}^i(\mathrm{G})$ is known, we can derive $\mathrm{H}^j(\mathrm{G})$ where $i < j$.

However, given $\mathrm{H}^j(\mathrm{G})$, it is not computationally feasible to find $\mathrm{H}^i(\mathrm{G})$ where $i < j$ because of the property of the hash function.

3.1.4 ID-based Authentication

ID-based authentication is described below which includes three phases: *set-up*, *key generation* and *key agreement*.

1. Set-up

According to the RSA algorithm [36], Trusted Center (TC) generates and publishes (n, g, e) but keeps (p, q, d) secret.

2. Key generation

For an authorized user A whose identification information is ID_a, TC computes $\mathrm{s}_a = \mathrm{ID}_a^{-d}$ (mod n).

For an authorized user B whose identification information is ID_b, TC computes $\mathrm{s}_b = \mathrm{ID}_b^{-d}$ (mod n)

Then TC issues $(n, g, e,$ ID_a, $\mathrm{s}_a)$ to user A and issues $(n, g, e,$ ID_b, $\mathrm{s}_b)$ to user B.

3. Key Agreement

Step 1: A generates the secret random R_A, A calculates

$$T_A = g^{R_a+ID_b} s_a(\mathrm{mod}n) \tag{3.1}$$

B generates the secret random R_B. B calculates

$$T_B = g^{R_b+ID_a} s_b(\mathrm{mod}n) \tag{3.2}$$

Step 2: B sends A: T_B

Step 3: A computes

$$K_{AB} = ((g^{-ID_a}T_B)^e ID_b)^{R_A} = g^{eR_AR_B} \tag{3.3}$$

Step 4: A sends B: T_A

Step 5: B computes

$$K_{AB} = ((g^{-ID_b}T_A)^e ID_a)^{R_B} = g^{eR_AR_B} \tag{3.4}$$

3.2 Contributory Group Key Establishment and Management Schemes

Several group key schemes have been proposed already. They can be classified broadly into two categories, namely, centralized [42], [49], [65], [68], [69] and contributory [3], [4], [5], [6], [7], [10], [22], [24], [26], [27], [43], [54], [55], [56], [67]. By extending the integer Diffie-Hellman key exchange (DH), several contributory group key agreements have been proposed which include Ingemarsson et al. for teleconferencing (ING) [22], Burmester-Desmedt (BD) [10], Skinny TRee (STR) [26], [56], Group Diffie-Hellman (GDH) [5], [6], [55], Octopus [7], Tree-based Group Diffie-Hellman (TGDH) [25], Distributed Logical Key Hierarchy (D-LKH), and Distributed One-way hash Function Tree (D-OFT). They are introduced in this section. Before the execution of the following group key agreements, every group member M_i should agree on the generator, α, an integer. Each group member, M_i, generates a session random integer, namely, r_i, and calculates the blinded key α^{r_i}, in advance.

3.2.1 Burmester-Desmedt (BD)

The BD protocol is one of the earlier group key schemes which extend DH scheme. The group key can be calculated in three rounds.

Table 3.1: Burmester-Desmedt (BD) Protocol

Protocol 1 - Group Key Establishement: BD
Round *1*: Broadcast the partial keys Every group member, M$_i$, broadcasts: $Z_i = \alpha^{r_i}$
Round *2*: Calculate key material Every group member, M$_i$, computes $X_i = (Z_{(i+1) \bmod n}/Z_{(i-1) \bmod n})^{r_i}$
Round *3*: Compute the group key Every group member reaches the group key: $k = Z_{i-1}^{nr_i} \bullet X_i^{n-1} \bullet X_{i+1}^{n-2} \bullet\bullet\bullet \bullet X_{i-2} \bmod p$ $= \alpha^{N_1N_2+N_2N_3+\cdots+N_nN_1} \bmod p$

However, when a new group member joins or when a group member leaves, most of the group members need to refresh the random session r$_n$ and follow the steps mentioned above one by one. Therefore, this scheme requires high resources.

3.2.2 Octopus Protocol

The Octopus protocol is one of the earlier tree-based contributory group key agreements organized with virtual hierarchy. In this scheme, all group members are divided into four subgroups, namely, A, B, C and D. In every sub-group, there is one group member leader available, namely M$_A$, M$_B$, M$_C$ or M$_D$, respectively. The role of every group leader is to collect the contributions of every sub-group member and calculate the intermediate value, R_A (or R_B, R_C, R_D,). Then, the four group leaders launch the DH scheme to compute group key G and send G back to every sub-group member. Specifically, the group is split and the intermediate values are computed as below (r$_i$ is the contribution of group member M$_i$): Subgroup A includes group members $M_1 \cdots M_{n/4}$; the leader of sub group A, calculates $R_A = \Pi_{1<i<n/4} r_i$; Subgroup B includes group members $M_{n/4+1} \cdots M_{n/2}$; the leader of sub group B, calculates $R_B = \Pi_{n/4+1<i<n/2} r_i$ Subgroup C includes group members $M_{n/2+1} \cdots M_{3n/4}$; the leader of sub group C, calculates $R_C = \Pi_{n/2+1<i<3n/4} r_i$ Subgroup D includes group members $M_{3n/4+1} \cdots M_n$; the leader of sub group D, calculates $R_D = \Pi_{3n/4+1<i<n} r_i$ Notice that the sub-group leader, for example, A, should has a secure channel between every other subgroup member. Via these channels, A can obtain r$_1$, $\ldots$, r$_{n/4}$ one by one. The same applies to B, C, or D. After the completion of the procedures mentioned earlier, group key G can be computed as below:

1st round: M$_A$ and M$_B$ exchange α^{S_A} and α^{S_B}.Both of them compute S$_{AB}$ = $\alpha^{S_A S_B}$

M$_C$ and M$_D$ exchange α^{S_C} and α^{S_D}. Both of them compute S$_{CD}$= $\alpha^{S_C S_D}$

2nd round: M$_A$ and M$_C$ exchange $\alpha^{S_{AB}}$ and $\alpha^{S_{CD}}$. Both of them compute G = $\alpha^{S_{AB} S_{CD}}$.

M$_B$ and M$_D$ exchange $\alpha^{S_{AB}}$ and $\alpha^{S_{CD}}$ Both of these compute G = $\alpha^{S_{AB} S_{CD}}$ The sub group leader, for example, A, sends every sub group member M$_i$ the partial key material Gα^{-r_i}. Every group member can calculate group key G. The same applies to B, C, or D.

3.2.3 Group Diffie-Hellman (GDH)

For a group key establishment scheme, two Group Diffie-Hellman key agreements [5], [6] GDH.2 and GDH.3, are proposed to generate the group key based upon an extension of the Diffie-Hellman two-party key exchange scheme. In this thesis, only the latter is considered, since it is more efficient than GDH.2.

When a new group member, M$_{n+1}$ joins, group member M$_n$ refreshes its random session r$_n$ with r$_n$'. Then M$_n$ calculates $n-1$ partial keys, $\alpha^{(\prod_{k=1}^{n-1} r_k) r_n'}$ and forwards the output to new member M$_{n+1}$ who works as the group controller in this round. The rest is the same as that for round n and $n+1$ of GDH.3. When group member M$_i$ leaves, the group controller, M$_c$, removes the partial keys which include the contribution of r$_i$, refreshes its random session r$_c$' and broadcasts the updated $n-1$ partial keys. All group members can update their group keys based upon the broadcast messages. For details please refer to Ateniese Steiner and Tsudik, [5], [6].

3.2.4 Tree-based Group Diffie-Hellman (TGDH)

The TGDH group key management scheme [25] integrates a binary tree structure with the intermediate sub-group key calculation formula of the *hypercube & Octopus* protocol [7]. TGDH assumes that all group members are sorted in a logical tree. In the following key tree, the sponsor selection policy and the group key update algorithm are introduced. A binary tree **T** is a key tree in which every node can be denoted as $< h, i >$ where h is the height (level) of the node and i is the index of the node at level h. Thus, every node is identified uniquely. There are two kinds of nodes in **T**, the leaf node and the intermediate node. Each leaf node in the tree represents a group member M$_i$. The intermediate node has two children. It represents a sub-group which includes group members who are represented by all of this intermediate node's offspring leaf nodes. Each node, in the binary tree, has two keys, *node key*, K and *blinded key*, BK. The node key associated with node (l, v) is K$_{<l,v>}$ and its blinded key $BK_<l, v> = \alpha^{K_{<l,v>}}$. Each leaf node's node key is a random integer r$_i$, which is generated by group member M$_i$. The intermediate node's node key is treated as the *sub-group key.* Without loss of generality, the root node's node key is called the *group key.* For each internal node $< l, v >$, its associated node key k$_{<l,v>}$ is derived from the keys of its two children, $< l+1, 2v >$ and $< l+1, 2v+1 >$, in the following manner:

The group key establishment scheme for TGDH is simple. The group members in the key tree, who are represented by the leaf nodes, exchange their blinded keys with their sibling nodes to perform the Diffie-Hellman key exchange scheme. This procedure is repeated at every level of the key tree until the root of the key tree is generated at round $log_2 n$ where n is the group size.

$$k_{<l,v>} = BK_{<l+1,2v+1>}^{k_{<l+1,2v>}} = {BK_{<l+1,2v>}}^{k_{<l+1,2v+1>}} = \alpha^{k_{<l+1,2v+1>} k_{<l+1,2v>}} \tag{3.5}$$

Table 3.2: GDH.3 Protocol

Protocol 2 - Group Key Establishment: GDH.3
Round *i* ($1 \leq i \leq n-2$): collect all contributions from every group member: M$_i$ sends M$_{i+1}$:$\alpha^{(\prod r_k \mid k \in [1,i])}$
Round *n-1*: Broadcast/multicast the partial key M$_{n-1}$ broadcasts M$_1$, M$_2$... M$_{n-2}$: $\alpha^{(\prod r_k \mid k \in [1,n-1])}$
Round *n*: Every group member factors out its own contribution r$_i$ by using the inverse r$_i^{-1}$ and sends the result to the last group member M$_n$:   M$_1$, M$_2$... M$_{n-1}$ sends M$_n$: $\alpha^{(\prod r_k \mid k \in [1,n-1] \wedge k \neq i)}$
Round *n+1*: M$_n$ raises the messages it received with the power of r$_n$ and then broadcasts all results M$_n$ works as the group controller and broadcasts M$_i$: $\{\alpha^{(\prod r_k \mid k \in [1,n] \wedge k \neq i)} \mid i \in [1, n-1]\}$ Every member M$_i$ picks up the corresponding partial key and raises its power with its random session r$_i$. In the end, every member reaches the same group key, $\alpha^{(\prod_{k=1}^{n} r_k)}$.

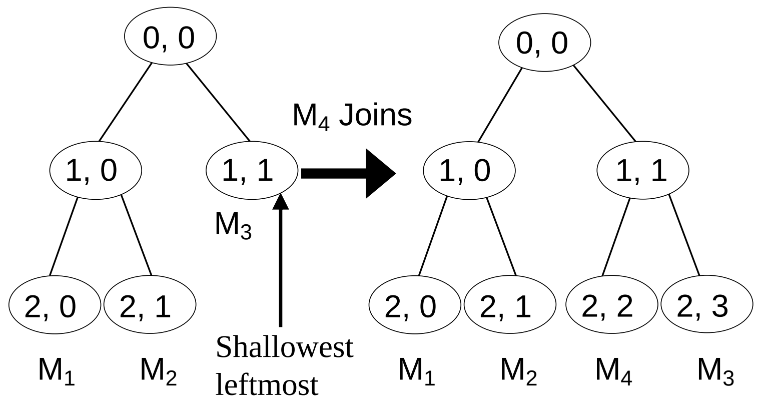

Figure 3.1: Sponsor for TGDH

When a group member joins/leaves, the TGDH protocol should select the sponsor node (in short, sponsor) which is in charge of the group key updates. When one group member joins the group, the shallowest leftmost leaf node in the key tree should be selected as the sponsor which will work as the sibling for the new group member. When one group member leaves, the sponsor should be the shallowest leftmost leaf node of the sub-tree rooted as the leaving member's sibling node. Then, all group members block except the sponsor which updates its own random secret, calculates the new keys on its key path based upon the new secret, and furthermore, broadcasts/multicasts the updated blinded keys on its key path. Finally, other group members can calculate the new group key based upon the updated blinded keys. For example, in Figure 3.1, M_4 joins. M_3 is selected as the sponsor. M_3 refreshes its random secret $K_{<2,3>}$ ($K_{<2,3>}$ is equal to r_3) and updates the node keys, { $K_{<1,1>}$, $K_{<0,0>}$} as well as the blinded keys, {$BK_{<2,3>}$, $BK_{<1,1>}$} on its key path. Finally, updated blinded keys {$BK_{<2,3>}$, $BK_{<1,1>}$} are broadcast/multicast. M_1,M_2 and M_3

are able to calculate the new group key $K_{<0,0>}$ based upon them.

3.2.5 Skinny TRee (STR)

As one of the earlier tree-based group key management schemes, Skinny TRee (STR) [56] is promoted by Yim, Perrig and Tsudik [26] to handle membership events. This scheme utilizes an unbalanced key tree in which every leaf node represents a group member. All intermediate nodes play a management role. Every group member M_i should generate a random secret r_i and calculate its leaf node's blinded key $BK_{<i,1>} = \alpha^{r_i}$. In the first round, every member broadcasts $BK_{<i,1>} = \alpha^{r_i}$ where $1 \leq i \leq n$ and n is the group size. In the k^{th} round, intermediate values $BK_{<k+1,0>} = BK_{<k,0>}^{K_{k,1}} = BK_{<k,0>}^{r_k}$ where $BK_{<2,1>} = \alpha^{r_1 r_2}$ will be broadcast to other members so that exponentiations can be raised. After n rounds, every member can reach the group key:

$$K_{<n,0>} = \alpha^{r_n \alpha^{r_{n-1} \cdots \alpha^{(r_1 r_2)}}} \tag{3.6}$$

To handle group members joining, STR adds a new leaf node to represent the new member. This new leaf node is treated as the current root's sibling and a new root node is created which works as the former root and the new member's parent. The group member representing with the leaf node right below the new leaf node is selected as the sponsor. When a group member leaves, the leaf node representing the leaving group member and the corresponding sibling node are deleted. The group member represented by the leaf node right below the leaving member's leaf node is treated as the sponsor. What the sponsor performs is the same for both the joining and leaving protocols. The sponsor refreshes its random secret and updates all node keys and blinded keys associated with the nodes above it. Finally, the updated blinded keys are multicast and every other group member can calculate the new group key. STR shows computational and communication efficiency for group membership additions but not always for group membership deletions. STR reduces the number of rounds needed to update the group key as compared with TGDH.

3.2.6 Distributed One-way hash Function Tree (D-OFT)

Like TGDH, Distributed One-Way hash function Tree (D-OFT) utilizes the binary key tree to support a group generation and management. Every leaf represents a group member and intermediate nodes work as the key management role. Every node in the key tree hosts the node key and a blind key. The blind key is calculated as below:

$$K_B = g(K) \tag{3.7}$$

where K: node key; K_B : blinded key; g: one way hash function;

The leaf node's key is generated by the group member and the intermediate node key is computed according to the formula below:

$$K_i = f(g(K_{left\_child(i)}, K_{right\_child(i)})) \tag{3.8}$$

where f: mix function to mix together the two parameters.

Like TGDH, the group key is the root node's node key. To calculate the group key, every leaf member should know all node keys on its key path and all blinded keys on its sibling key path. Following the formula mentioned earlier, the group key can be computed by every group member. Whenever group members join/leave, new contributions of the changing member's sibling should be refreshed and the corresponding node key and blinded key on its key path should be updated. This solution assumes a secure channel between the group members in advance so that the updated node key and blinded key can be encrypted and sent to its counterpart via the secure channel. Compared with TGDH, this method is computationally efficient since the one-way hash function rather than the exponential operation is used to calculate the blinded key and node key (including the group key). However, assuming the pre-deployed secure channel between group members is a limit for the installation of this proposal.

3.2.7 Distributed Logical Key Hierarchy (D-LKH)

Distributed Logical Key Hierarchy (D-LKH) is proposed without the supporting of a centralized server. In this scheme, every group member plays a symmetric role. This solution utilizes the logical tree which has left tree L and right tree R. Both L and R include a set of group members. Every group member in L agrees on a shared key K_L , and those in R, a shared key K_R. Member M_L and M_R work as the group leaders of L and R respectively. To share a mutual key between groups L and R, the steps described below are required:

1. M$_L$, the group leader of L, chooses a new key K$_{LR}$ and sends it to the group leader of R, M$_R$, via a secure channel.

2. M$_L$ encrypts K$_{LR}$ with key K$_L$ and multicasts the ciphertext to all the group members in L. M$_R$ encrypts K$_{LR}$ with key K$_R$ and multicasts the ciphertext to all the group members in R.

3. All members within L and R receive the ciphertext and decrypt the group key, K$_{LR}$. As with D-OFT, this solution assumes a secure channel between the group members in advance.

3.3 Distributed Algorithm to Forward Partial Keys

Several distributed algorithms have been proposed to deliver partial keys for previous contributory group key establishment schemes.

3.3.1 Connected Dominating Set for GDH (CDS-GDH)

In GDH.2 and GDH.3, communication costs between nodes are not considered and group members are assumed to be sorted before the launching of the group key establishment schemes. Li, Wang and Frieder [62] propose a Connected Dominating Set (CDS) to divide the group into subgroups. They recommend a hierarchical key architecture where every subgroup generates its sub-group key and all dominators share another group key. To relay intermediate GDH messages over dominators, a spanning tree is constructed and a post-order walking algorithm is deployed. The post-order walking algorithm can satisfy GDH which assumes a path which covers all group members. The number of rounds for the post-order walking algorithm on a tree is $2n$.

3.3.2 Shortest Path for GDH (SP-GDH)

Anton and Duarte [4] find that, in GDH.2 and GDH.3, predefined node sequences may not be the best choice for network nodes with geographic placements. They propose a method to generate the node sequence with the assistance of node location information. The path's end point broadcasts a request to search for the next node. The first responding node is added to the end of the path. This procedure is repeated until all group members are involved in the path. This method can be treated as the Shortest Path problem, with the number of rounds, O (n).

3.4 Centralized Group Key Schemes

Several centralized group key schemes have been proposed to achieve performance efficiency. Since this thesis proposal focusses mainly on the contributory group key schemes which are utilized in resource-limited networks, this sub-section gives a brief review of some key tree-based centralized schemes which are relevant to the proposed hybrid architecture. Logical Key Hierarchy (LKH) [65] uses a Key Distribution Center (KDC can be viewed as a key server or group controller) to maintain a key tree, with which individual keys and auxiliary keys are associated. The KDC establishes a secure channel between every group member and itself via an individual key. Intermediate auxiliary keys which are associated with the intermediate nodes in the key tree can be forwarded to every group member after being encrypted with the individual key. When the membership updates, new auxiliary and new group keys are generated by the KDC and they are encrypted with the individual key known by a group member and the KDC. The KDC forwards these ciphertexts to corresponding group members one by one. In turn, every group member can decrypt them using the individual key it owns. The rekey messages will contain at most $2log_2n$ keys.

Efficient Large-Group Key [42] (ELK) uses a hierarchical key tree as well. This novel and sophisticated agreement is applicable to large groups, refreshing its key tree in periodic intervals for the joining group members.

Keygem [68], [69] follows a periodic batch rekey strategy, in which all members should have their time [68] synchronized and agree on a rekey period. Furthermore, they use a marking algorithm to handle the *J Join /L Leave request* (where J members join and L members leave).

3.5 FEA-M and Improved Variants

Securing real-time multimedia data is a challenging task since the size of the data is usually very large and the data needs to be processed in a short time interval. Standard cryptographic algorithms will usually result in a large overhead, rendering them inefficient.

Yi, Tan, Siew, and Syed [70] have proposed a novel algorithm called Fast Encryption Algorithm for Multimedia (FEA-M) which requires only 1.5 XOR operations to encrypt one bit of plaintext. This is significantly less compared to other encryptions such as Rijndael, Crypton, Twofish, Cast256 and Serpent [32], [70], [28]. FEA-M is based upon the Boolean matrix theory which involves matrix addition and multiplication over the finite field GF(2)= 0, 1. FEA-M's security is based upon the complexity of solving non-linear equation groups and variable linear equation groups. To protect the key material against both passive and active attacks, an ID-based key agreement is utilized to secure FEA-M's key exchanges [69].

Mihaljevic and Kohno [33], [34] analyze FEA-M's security and find it is not secure enough when the first plaintext blocks are all 0s. Furthermore, Mihaljevic indicates that FEA-M cannot work if one ciphertext package is lost during transmission. He proposes an improvement to counter this vulnerability [35].

3.5.1 FEA-M and the Improved Variants

FEA-M uses an ID-based Diffie-Hellman key agreement protocol to generate a common secret key, k, an integer, between the sender and the receiver [69]. Based upon the value of k, FEA-M generates a common key matrix $\mathbf{K}$ and a common initial matrix $\mathbf{V}_0$ which are binary matrices of order n. The details of the algorithm for generating $\mathbf{K}$ and $\mathbf{V}_0$ have been published previously [69], [70].

The plaintext message is divided into a series of blocks, P_1, $\mathrm{P}_2 \ldots P_r$, with the same length, n^2, where n is 64 and r is an integer [70]. If the length of the last block is less than n^2, it is padded with 0s to make its length n^2. Each plaintext matrix, $\mathrm{P}_i (1 \leq i \leq r)$, is encrypted into a ciphertext matrix C_i and each corresponding ciphertext matrix C_i is decrypted into a plaintext matrix P_i according to formulas below:

$$C_i = K \bullet (P_i + C_{i-1}) \bullet K^i + P_{i-1} \tag{3.9}$$

$$P_i = K^{-1} \bullet (C_i + P_{i-1}) \bullet K^{-i} + C_{i-1} \tag{3.10}$$

$$P_0 = C_0 = V_0 \tag{3.11}$$

The vulnerability of FEA-M has been identified and improvements have been proposed. Mihaljevic and Kohno [33], [34] point out that the real uncertainty about the secret key of FEA-M is undesirably smaller than expected since the effective secret key size, under realistic known and chosen plaintext attacks, is much smaller than the nominal one. It occurs while the first set of blocks is all 0s. They conclude that when the key is a 64*64 matrix, the nominal secret key size is 4096 bits but the effective secret key size is only 134 bits.

Furthermore, Mihaljevic [35] indicates that if one ciphertext block is lost during transmission, subsequent ciphertext blocks cannot be decrypted since they depend upon former ciphertext blocks. To overcome this weakness, he proposes a new encryption algorithm, which is described by the formulas below:

$$C_i = K \bullet (P_i + K \bullet V \bullet K^i) \bullet K^{i+n} + K \bullet V \bullet K^i \tag{3.12}$$

$$P_i = K^{-1} \bullet (C_i + K \bullet V \bullet K^i) \bullet K^{-(i+n)} + K \bullet V \bullet K^i \tag{3.13}$$

Therefore, according to Mihaljevic's improvement [35], if C_i is a lost block, no further impact on subsequent blocks occurs.

Chapter 4

Contributions

4.1 Overview and Assumptions of Contributions

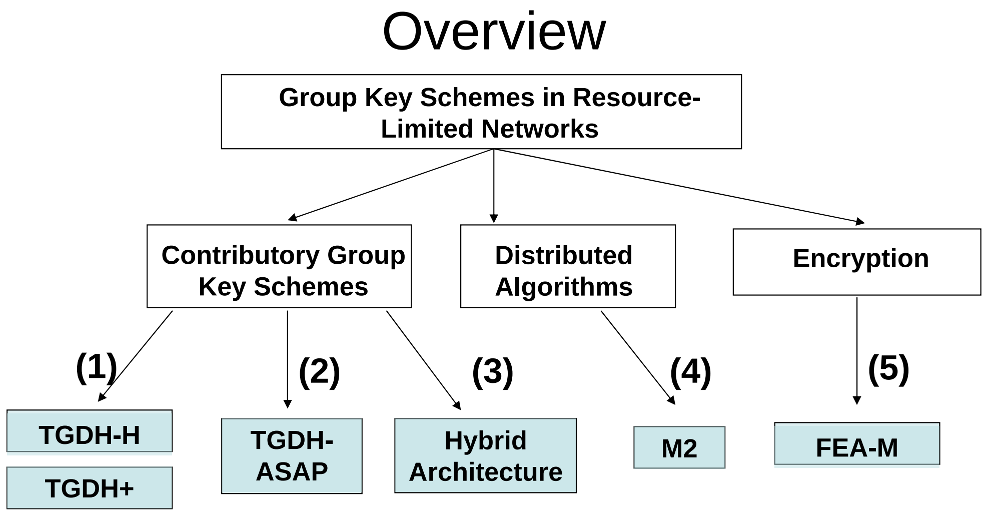

Figure 4.1: Outline of Contributions

4.1.1 Overview

Group communication systems demand efficient and fault-tolerant group key schemes to secure the services they provide. Several group members, for example, cooperate with each other to achieve a rescue task in malicious environments such as battle-fields. This group comprises some low-end mobile clients and a few high-end clients as well. We call the former the node and the latter the server. They are able to set up and enable the group communication service via deploying Liu's routing protocol [78] to enable the MANET. The security solution proposed in this thesis is the group key scheme. Therefore, a group key should be refreshed to guarantee that only the insider holds the group key. Every group member may face such scenarios: 1) The group key should be established at a certain time. 2) During the group session, new group members may join and some members may leave the group at anytime. 3) The group can be portitioned or sub-groups can merge together again. Group key schemes should be proposed to satisfy each sub-group. 4) The key server cannot always be on-line because of failure or battery depletion. Therefore, methods should be proposed to establish/manage the group key schemes to satisfy the requirements for the scenario mentioned above.

In Figure 4.1, an overview is presented of the five contributions of this thesis to satisfy the requirements listed above.

The first contribution is two efficient individual group key management schemes, namely, Tree-based Group Diffie-Hellman with one-way Hash function (TGDH-H) and Tree-based Group Diffie-Hellman Plus (TGDH+), which are suitable for scenarios with a low rate of group member join/leave requests.

The second contribution is the periodic group rekey scheme, TGDH Amortized Signing Amortized Path (TGDH-ASAP), which satisfy scenario with a high rate for group member join/leave.

The third contribution is maximum matching algorithms (M2) which are used to support the partial key delivery for group key establishment scheme.

The fourth contribution is a hybrid group key management architecture which accommodates a centralized and contributory group key scheme. This solution can be deployed for cases in which the key server is switching from online to offline.

The fifth contribution is the improved FEA-M encryption algorithm which can be used to encrypt group messages, especially for multimedia data, while using the group key.

4.1.2 Assumptions

Support of Group Communication System (GCS)

Our first assumption is that the group key schemes (transport component) are developed with the support of an underlying GCS [3] [29] which are achieved via reliable and ordered multicasts and unicasts. To achieve this task, the certain services are required:

Every group member should be notified with each group membership change. As with previous GCSs [3], [18], [49], [51], the group membership update event is treated as the *view*. This research requires that group members should be able to observe the same set of messages between two sequential group membership events. Furthermore, the message order for the sender's request must be preserved. We say that the *Virtual Synchrony* service [18] is achieved if both of them are provided.

After receiving all necessary rekey messages, every group member can re-calculate the new group key. In practice, this requirement needs the *Virtual Synchrony* service as well.

In a word, this proposal assumes that the data transport of the GCS should be reliable and that the message delivery should be ordered. This can be achieved via reliable multicast protocols [80]. In fact, several GCSs have been deployed already. For example, in Secure Spread [3], while the group membership change happens, the GCS robust algorithm informs the application that a *new view* should be installed. The application, in return, sends the ACK following the set of *old view* messages. After this step, the application is blocked until the *new view* is forwarded. Since this system satisfies the *Sending View Delivery* property which requires messages to be forwarded in the *same view*, the receiver can share the same set of views as the sender. Therefore, in the Secure Spread system, *Virtual Synchrony* semantics can be preserved by providing a reliable and ordered guarantee within *a view.* The Secure Spread system has proven to be resilient to any finite sequence of events including cascading ones. Furthermore, Secure Spread is robust enough to handle process crash events. For details about the system design and algorithm implementations please refer to Ateniese, Steiner and Tsudik *New multiparty authentication services and key agreement protocols* [3].

Several previous popular group key schemes such as TGDH [25], STR [26], and Queue-Batch [41] assume the availability of GCSs.

Authentication and Certification

In the purpose of this thesis certification is assumed to have been deployed/installed for every group member/node in advance since this technology has been mature for quite some time. Sample solution can be Computer Kerb, X.509, ID-based authentication, etc. In this thesis, I describe ID-based authentication as an example in section 3.1.4. Menezes, Oorschot and Varstone have provided other solutions [32].
In this thesis, Partial keys are signed with RSA algorithms for multicast or hashed with the MAC method for unicast before forwarding to other group members. For details, please refer to the Modified Signature Amortization Information Dispersal Algorithm (M-SAIDA) in Algorithm 2 (Table 4.4).

Performance requirement

As with other popular software such as Unix, Linux, or Windows, it is better to be aware of the clients minimum requirements in terms of the device's system demands with regard to processor and memory capability. The proposed group key schemes are not applied to devices with lower-end resources. For details, please refer to the chapter 5 performance assessments.

4.2 Contribution # 1: Efficient Individual Rekeying Schemes

Individual rekey schemes are critical for group communication with strict security requirements in which the group key should be refreshed for every group membership change. In this part, two new individual rekey proposals are demonstrated which are not only efficient but cannot compromise any security requirements.

4.2.1 Overview of the First Contribution

The individual rekey strategy provides strict backward secrecy and forward secrecy. Therefore, the individual rekey strategy can be deployed in financial/military applications which require strict security services.

Overview

In this section, two efficient individual group key management schemes are proposed. They both are enhancements of the TGDH group key management schemes which both include join/leave/merge/partition protocols. The key ideas in the proposal are listed below.

TGDH-H presents a number of extensions for TGDH by incorporating TGDH and an one-way hash function. To improve the efficiency of TGDH, TGDH-H updates the group key via a one-way hash function (Hash) for joining group members and postpones the DH-based key updates until group members leave. Furthermore, when a group member leaves, TGDH-H proposes two more sophisticated techniques, namely, moving the child key tree and the dominating algorithm. The former is a method for dragging the child key tree from one position to another. The latter lets every group member be aware of which member is responsible for updating which overlapped intermediate nodes in the key tree. Both of these techniques can lessen the computational cost and reduce communication overhead.

To enhance the efficiency of TGDH-H, TGDH+ is proposed. Unlike TGDH-H which updates all node keys and blinded keys associated within all key paths when a group member leaves, TGDH+ simply updates these node keys and blinded keys in the child key tree. This method reduces the number of exponential operations from O(n) to O(m) where n is the size of the total group, m is the size of the child key tree, and $m < n$.

Notice that the deployment of a one-way hash function may introduce a vulnerability in which an outsider with an illegally obtained group key can deduce the following group key until a group member leaves. Although this method introduces this small vulnerability contrary to strict security, the trade-off is worthy it because of significant performance improvements.

Concepts and notion

Before describing the proposal, some background concepts will be introduced [25].

Key path: A path starting at the leaf node hosted by a group member, e.g. M$_i$, and ending at the key tree's root. The key path is named KP_i. M$_i$ hosts all node keys on KP_i, namely, KEY_i^*.

Sibling path: Sibling nodes corresponding with nodes on group member M$_i$'s *key-path* constitute M$_i$'s *sibling path.* M$_i$ hosts all blinded keys on its *sibling path*, namely, $BKEY_i^*$.

Key sub-path: a sub-path starting at any node, N$_x$ and ending at any other node, N$_y$ on a key path is called a key sub-path, namely, $KSP_{i,xy}$. All node keys on *key sub-path* are called KEY_i^*, x, y.

|| denotes the concatenation.

Notice that, with the support of group communication systems, every group member can be aware of the same set of join/leave requests. Hence, utilizing the same strategy and being aware of the same change of membership, every group member can get the same key tree structure. Additionally, as with TGDH, every group member is required to maintain the node keys associated with the nodes on its key path and the blinded keys associated with the nodes on its sibling key path.

4.2.2 TGDH-H

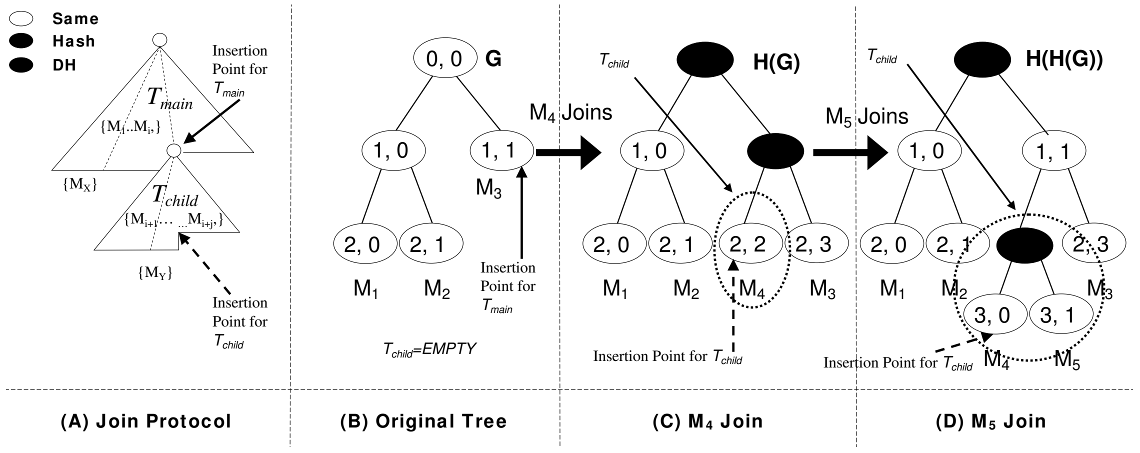

Figure 4.2: TGDH-H: Key Tree Updates for Group Members Joining

In this sub-section, an efficient group key management scheme (TGDH-H) is proposed. Utilizing a hash function to handle group member joining has been suggested by some centralized group key management schemes such as ELK [42] and LKH+ [61]. To date, contributory schemes have not adopted this technique. In the following section, four basic protocols for TGDH-H — join, leave, merge and partition — are described.

Join Protocol

1. Method to update the Key Tree Structure

The key tree shown in Figure 4.2 (a), includes two parts: the main key tree, T_{Main} and a child key tree, T_{child}. At the very beginning of the group key scheme, both of them are empty which means that there are no nodes available. Every key tree should have its insertion point, which is the shallowest leftmost node in the key tree. For every group membership change, the rules below should be followed: 1) When a group member leaves or the group partitions/merges, T_{child} will merge into the key tree, T_{main}, and then T_{child} is assigned as *EMPTY*. 2) When a group member joins, the method of inserting it into the key tree should be based upon whether T_{child} is *EMPTY*. To check whether there exists T_{child}, every node should allocate a pointer variable pointing to the root of T_{child}. If the pointer is NULL, T_{child} is empty. If T_{child} is not *EMPTY*, the new group member should be appended to the T_{child}. Otherwise, T_{child} should be generated with its root located at the insertion point of T_{Main}. Then, T_{child} is not EMPTY. The remaining new join nodes should be appended into T_{child} and located at the insertion point of T_{child}. Figure 4.2 (a) – (d) shows a scenario in which group members ($M_1 \ldots M_i$) are already within the group and, then, the following group membership events happen:

$$< M_x^{Leave}, M_{i+1}^{Join}, M_{i+2}^{Join} ... M_{i+j}^{Join}, M_y^{Leave} | where \quad j \geq 0\rangle$$

Between the two leave requests from M_x and M_y ($1 \leq x \leq i; 1 \leq y \leq i + j$), group members $M_{i+1}, \ldots M_{i+j}$ request to join one by one. Notice that this event model can represent all scenarios occuring in group membership changes. That is because all event sequences can be segmented by leave events. For the remainder of this thesis, this model will be utilized to demonstrate group events.

With the group membership change input, T_{child} should be *EMPTY* after M_x leaves. Then, when M_{i+1} requests to join, the join protocol generates T_{child} with the root located at the T_{Main} insertion point and the join protocol inserts M_{i+1} into T_{child}. Now T_{child} is not *EMPTY.* Subsequent join requests, $M_{i+2}, \ldots M_{i+j}$ can be appended into T_{child} at T_{child}'s insertion point. After M_y leaves, T_{child} is assigned to *EMPTY.*

Here are two examples. The tree shown as Figure 4.2 (b) is the beginning scenario. The trees shown in Figure 4.2 (c) and Figure 4.2 (d) result from the joining of M_4 and M_5, respectively. Specifically, as shown in Figure 4.2 (c), M_4 joins and a new leaf $< 2, 2 >$ is generated to represent it. The insertion point for T_{Main} is located at node $< 1, 1 >$ which should be renamed $< 2, 3 >$ and works as the sponsor. Therefore, a new intermediate node $< 1, 1 >$ is generated which works as both sponsor $< 2, 3 >$ and the new leaf $< 2, 2 >$'s parent. As shown in Figure 4.2 (d), M_5 joins and a new leaf $< 3, 1 >$ is generated to represent it. $< 3, 1 >$ is appended into T_{child} rooted with $< 1, 1 >$. Node $< 2, 2 >$, representing member M_4, is selected as the sponsor and is renamed as $< 3, 0 >$. The join protocol generates a new node $< 2, 2 >$ which works as $< 3, 0 >$ and $< 3, 1 >$'s parents.

2. Group key updates

For a group member join request from $M_{i'}$, the proposed join protocol selects the sponsor S in the same manner as TGDH. However, the difference between TGDH and the proposed approach is that every group member updates the current group key, G with $\mathbf{H}$(G) rather than updating all keys associated with the nodes on sponsor S's key path, where $\mathbf{H}$ is a secure one-way hash function. Then, S and M_i, initiate a 2-party DH key exchange scheme to generate the shared key, K, which works as the node key of S and M_i,'s parent node. Finally, S sends M_i, the encrypted current group key, {H(G)}K and M_i, decrypts the ciphertext with key K to obtain the current key, H(G).

For example, in Figure 4.2 (c), M_3 is selected as the sponsor. It refreshes its secret random r_3 with r_3' and calculates the updated blinded key of its leaf node, $BK'_{<2,3>} = \alpha^{r_3'}$. Then M_3 and the new group member M_4 launch a 2-party DH to calculate a shared key,$K_{<1,1>}$. M_3 sends $BKEY_3$* || $BK'_{<2,3>}$|| $\{\mathbf{H}(G)\}K_{<1,1>}$ to M_4. M_4 calculates $K_{<1,1>}$ and decrypts the ciphertext $\mathbf{H}$(G). Other members can calculate the new group key, $\mathbf{H}$(G), via a secure hash function since they know the current group key, G.

In Figure 4.2 (d), M_4 is selected as the sponsor. It refreshes its secret random r_4 with r_4' and calculates the updated blinded key of its leaf node, $BK'_{<3,0>} = \alpha^{r_4'}$. Then M_4 and the new group member M_5 launch a 2-party DH to calculate a shared key, $K_{<2,2>}$. M_4 sends $BKEY_4$* ||$BK'_{<3,0>}$||$\{\mathbf{H}(\mathbf{H}(G))\}K_{<2,2>}$ to M_5. M_5 calculates $K_{<2,2>}$ and decrypts the ciphertext $\mathbf{H}(\mathbf{H}(G))$.

Notice that the mutual authentication between the sponsor and the new group member will deploy previous mature technologies such as certifications or the ID-based authentication described in section 3.1.4.

Leave Protocol

1. Strategy for updating key tree structure

Suppose that group member M_i, who is represented by the leaf $< h, i >$, leaves the group. Figure 4.3 shows the outline of the leave protocol for TGDH-H.

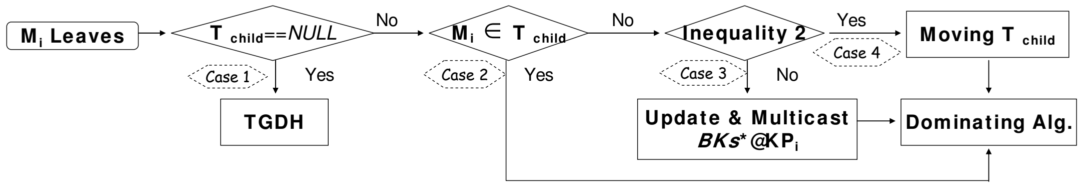

Figure 4.3: TGDH-H: Leave Protocol

If T$_{child}$ is not available, call it *case 1*. The leave protocol is as same as that for TGDH. If T_{child} is available and $< h, i >$ is within T_{child}, call it *case 2*. The key tree structure stays the same. If T_{child} is available and $< h, i >$ is not within T_{child}, there are two cases: either moving T_{child}, which is shown in Figure 4.4 (a), or not. The leaf node $< h, i >$'s position and computational cost decide whether T_{child} is moved. Inequality (4.1) decides which is more efficient, moving T_{child} or not. The left side of inequality (4.1) demonstrates the computation cost for updating keys associated with all the nodes in T_{child} and all the nodes in key path KP$_j$ (starting at the root of T_{child} and ending at the root of the key tree) in the case in which T_{child} is moved. When T_{child} is not moved, the right side gives the computation costs. It includes the computational cost to update keys associated with all nodes in T_{child}, with the key sub path KP$_j$ (starting at the root of T_{child} and ending at the root of the key tree) and with the key path KP$_i$ (the key path of the leaving group member M$_i$).

$$N^{Expon.}_{T_{Child}+KP_j} > N^{Expon.}_{KP_j} + N^{Expon.}_{KP_i+T_{Child}} \tag{4.1}$$

where N^y_x *is the number of y operations for all members within x.*

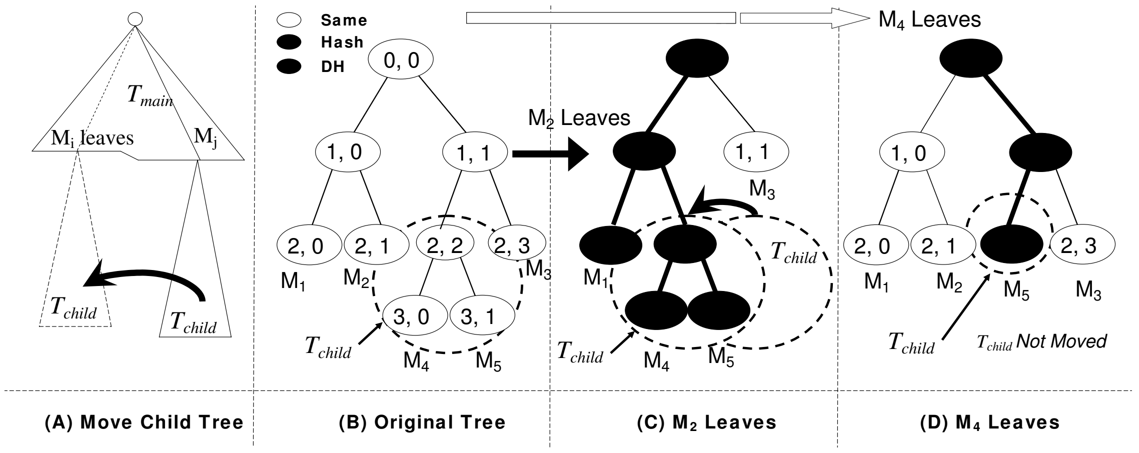

Figure 4.4: TGDH-H: Key Tree Updates for Group Members Leaving

If moving T_{child} can result in a performance improvement (inequality (4.1) is false), T_{child} should be moved to take $< h, i >$'s position and $< h, i >$ is cut off. This scenario is called *case 3*.

Otherwise, (inequality (4.1) is true, and T_{child} stays the same), it is called *case 4*.

For example, Figure 4.4 (b) is the original key tree. Figure 4.4 (c) shows that M$_2$ leaves. Since M$_2$ is not within T_{child} and what is more, inequality (4.1) is false, T_{child} rooted at $< 2, 2 >$ is moved to replace the position of node $< 2, 1 >$ to obtain the performance improvement. The former node $< 2, 1 >$ is cut off. As its left child node is removed, node $< 1, 1 >$ is deleted. $< 1, 1 >$'s right node $< 2, 3 >$ is renamed $< 1, 1 >$ and it is promoted to its parent's position.

Figure 4.4 (d) demonstrates that when M$_4$ leaves, T_{child} need not be moved since M$_4$

is within T_{child}.

2. Group key updates

To update the group key in the case in which a group member leaves, the leave protocol should handle case 1 to 4 separately.

Case 1:

As showed in Figure 4.3, the leave protocol is as same as that for TGDH.

Case 2, 3 and 4:

To update the group key in the case in which a group member leaves, the leave protocol should update all the node keys and blinded keys associated with the nodes in key paths which have one or more nodes added/deleted.

Obviously, the node key and the blinded key of every node within T_{child} should be updated. So do all keys on the leaving member's key path and on the T_{child}'s key path. The *dominating algorithm* is used to handle this situation and it is described at Table 4.1.

Dominating key path:

If two key paths intersect, we say that the right key path is dominated by the left key path. Therefore, the left key path is the dominating key path and is responsible for updating the overlapped nodes on the two key paths. For example, in Figure 4.4 (b), KP_4, the key path for M_4, intersects KP_5, the key path for M_5, at $< 2, 2 >$. Since KP_4 is to the left of KP_5, KP_4 dominates KP_5. Therefore, M_4 should update and multicast the blind keys for $< 2, 2 >$.

Without consideration for the root of the key tree, assume a key path KP$_i$ intersects $n-1$ other key paths, KP$_1$, KP$_2$ …KP$_{i-1}$, KP$_{i+1}$…KP$_{n-1}$, one by one from the leaf node to the root, where n is an integer and n is less than the height of the tree. Assume that the $n-1$ corresponding intersections are $< x_1, y_1 >$, $< x_2, y_2 >$ …$< x_{n-1}, y_{n-1} >$. The key path KP$_i$, is divided into the following n key sub-paths: $KSP_{i,<h,i>,<x_1,y_1>}, KSP_{i,<x_1,y_1>,<x_2,y_2>} ... KSP_{i,<x_{n-1},y_{n-1}>,<0,0>}$ The *dominating algorithm* (algorithm 1 at Table 4.1) describes how to update and forward the keys on the key sub-paths mentioned above.

Table 4.1: Dominating Algorithm

Algorithm 1: Dominating Algorithm	
Every Sponsor M_i :	
Step 1:	Update $KSP_{i,<h,i>,<x_1,y_1>}$
Step 2:	**IF** all updated blinded keys associated with key paths which are dominated by M$_i$ are send out repeat computing *node keys & blinded keys* on its key path until it cannot continue; multicast updated blinded keys on M_i 's key path; **ELSE** wait for updated blinded keys associated with key paths which are dominated by M_i; go to the beginning of step 2; **End IF**
All Group Members:	
Step 3:	After receiving the blinded keys from all sponsors, update the node keys on its key path.

The dominating algorithms treat fault nodes as the leaving node in the key tree. The key tree structure is modified following the leave protocol for TGDH. This means that the fault node and its parent node are deleted and the fault node's sibling node is promoted to its parent node. The outlines of the leave protocol for TGDH-H can be found in Table 4.2.

For example, in Figure 4.4 (c), after moving T_{child}, all keys associated with the nodes in T_{child} and T_{child}'s key path are updated:

1st round: Key path of M_5 is dominated by that of M_4. M_5 multicasts $BK_{<3,1>}$.

2nd round: M_4 multicasts $BK_{<3,0>}$, $BK_{<2,1>}$ and $BK_{<1,0>}$. In Figure 4.4 (d), all keys associated with the nodes within T_{child} and T_{child}'s key path are supposed to be updated:

3st round: Key path of M_5 multicasts $BK_{<2,2>}$ and $BK_{<1,1>}$.

Notice that the authentication to secure multicast messages will deploy the M-SAIDA described in Table 4.4.

Merge and partition protocols

When the group is divided into sub-groups, the partition protocol will treat the members who cannot be in contact with the group as leaving members. In this case, each group member will handle the *0 join & L leave* scenario. In a similar way, when sub-groups merge, the merging protocol deals with the *J join & 0* leave scenario. For every sub-group, the group member hosting the leftmost shallowest key path is treated as the sponsor for the sub-group which generates the new session secret key, updates keys on its key path and multicasts the updated keys. Both the merge protocol and the partition protocol can use algorithm 1: *Dominating Algorithm* to handle the *J join & 0 Leave* and *0 Join & L leave* scenario respectively.

Table 4.2: TGDH-H: Leave Protocol

Protocol 3 - Leave Protocol for TGDH-H	
Step 1:	$M_i \in \{M_1 \ldots M_n\}$represented by leaf $< h, i >$, request to leave.
Every Group Member:	

Step 2:	**IF** (T_{child} is not available)
	Leave protocol for TGDH.
	ELSE
	IF ($< h, i >$ is not within T_{child})
	IF (inequality (4.1) is FALSE)
	Delete T_{child}'s parent node
	Promote T_{child}'s sibling node to its parent node's position
	Move T_{child} to replace $< h, i >$'s position and cut off $< h, i >$
	Dominating Algorithm updates the keys in T_{child}
	ELSE
	The shallowest leftmost leaf of the subtree rooted with $< h, i >$'s sibling, S, works as the sponsor, too.
	Dominating Algorithm updates the keys in T_{child} and the key path of KP$_S$
	END IF
	ELSE
	Dominating Algorithm updates the keys in T_{child}
	END IF
	END IF

All Group Members:	
Step 3:	After receiving the blinded keys from all sponsors, update the node keys on its key path.

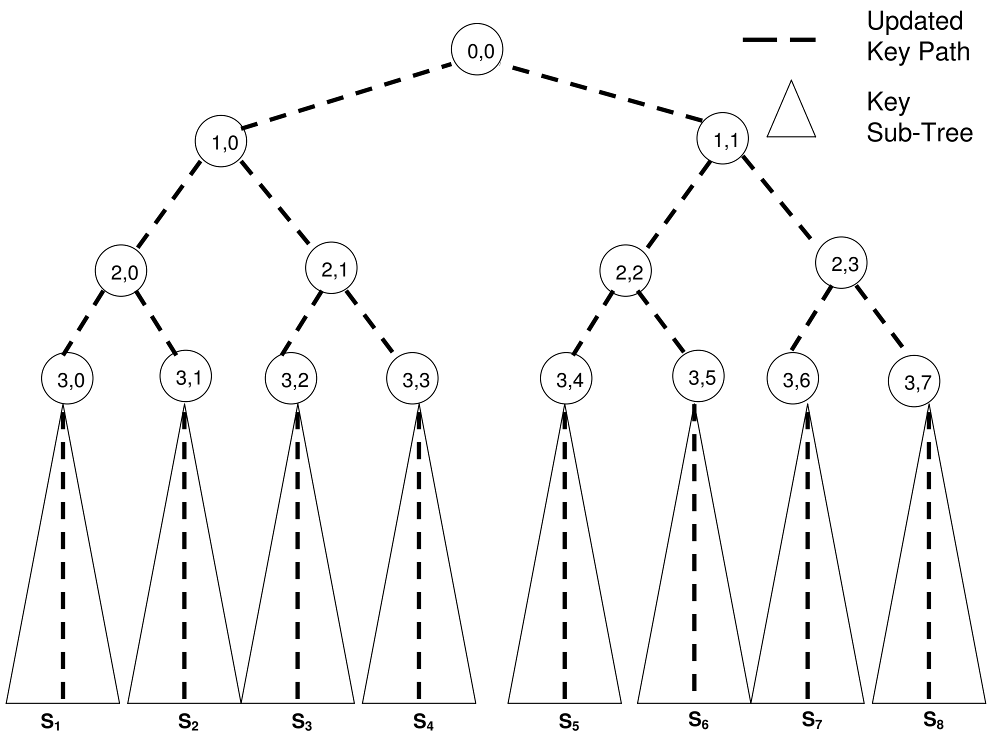

Figure 4.5: TGDH-H: Merge Protocol for 8 sub-groups

For example, the procedure to merge 8 sub-groups into a super group is shown in Figure 4.5. $S_1 \ldots$ and S_8 are selected as sponsors for the 8 sub-groups respectively. Using the dominating algorithm, the protocol can generate the group key within 3 rounds.

1st round: The key path for S_2 is dominated by that of S_1. The key path for S_4 is dominated by that of S_3. The key path for S_6 is dominated by that of S_5. The key path for S_8 is dominated by that of S_7. M_2, M_4, M_6 and M_8 update node keys and blinded keys on their key paths, respectively. Then, M_2, M_4, M_6 and M_8 multicast the updated blinded keys on their key sub path starting at the leaf node and ending at $< 3, 1 >$, $< 3, 3 >$, $< 3, 5 >$, and $< 3, 7 >$ respectively.

2nd round: The key path for S_3 is dominated by that of S_1. The key path for S_7

is dominated by that of S_5. Then, after calculating these node and blinded keys on their key paths, M_3 and M_7 multicast the updated blinded keys on their key sub path starting at the leaf node and ending at $< 2, 1 >$ and $< 2, 3 >$ respectively.

3th round: M_1 and M_5 update node keys and blinded keys on their key paths, respectively. Then, M_1 and M_5 multicast the updated blinded keys on their key sub path starting at the leaf node and ending at $< 1, 0 >$ and $< 1, 1 >$ respectively.

The partition protocol follows the same procedure. Notice that the merge and partition protocols for TGDH-H are the same as those for TGDH+. For simplification, the merge and partition protocols will not be introduced again in the later sub-sections. Furthermore, faults can occur even in join/leave/merge/partition protocols in the contributory group schemes. For joining/merging, the failure node is treated as a leaving member. The proposal simply treats them as members who leave. Then it is the leave/partition protocols' turn to handle them. The detailed procedure for leave/partition protocols follows what the leave/partition protocols do: deleting the leaving member's node and its parent node. The leaving node's sibling is promoted to its parent's position. The others functions in the same manner as described earlier. Notice that the authentication to secure multicast messages will deploy the M-SAIDA described in Table 4.4.

4.2.3 TGDH+

The TGDH+ group key scheme is an enhancement of TGDH-H. Like TGDH-H, TGDH+ also includes join, leave, merge, and partition protocols. The merge and partition protocols for TGDH+ are the same as those for TGDH-H. The join and leave protocols for TGDH+ both include two main parts: the key tree structure update strategy and the group key update. The strategies for updating the key tree structure for join and leave protocols are the same as those for TGDH-H. However, the method for updating the group key is different. In TGDH+, an auxiliary group key technique is introduced to update the group key efficiently. Specifically, for the join protocol, a method for adding an auxiliary group key is incorporated. When a group member leaves, in addition to the dominating algorithm and child key tree moving technique, this proposal includes an auxiliary group key technique which is used to deduce the future group key.

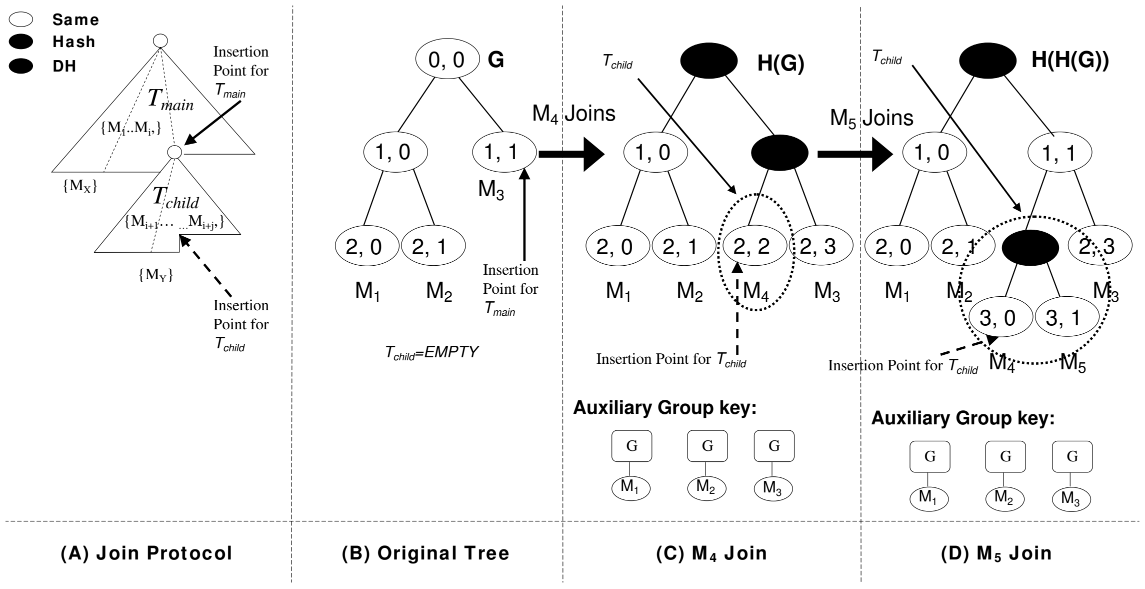

Figure 4.6: TGDH+: Key Tree Updates for Group Members Joining

Join Protocol

The join protocol for TGDH+ is almost the same as that for TGDH-H. In addition to what the TGDH-H join protocol does, the join protocol for TGDH+ stores an auxiliary group key for every group member in the main key tree as well.

Specifically, when a group member joins and the child key tree is *EMPTY*, the current group key G is stored by every group member within T$_{main}$ as the auxiliary group key G$_a$. When a group member joins and the child key tree is not *EMPTY*, the auxiliary group key stays the same. For example, the trees shown in Figure 4.6 (c) and Figure 4.6 (d) are what happens when M$_4$ and M$_5$ are inserted into the tree shown in Figure 4.6 (b). In Figure 4.6 (c), when M$_4$ joins, since T$_{child}$ is *EMPTY*, every group member in T$_{main}$, M$_1$, M$_2$ or M$_3$ stores group key G as its auxiliary group key: G$_a$ = G. In Figure 4.6 (d), when M$_5$ joins, since T$_{child}$ is not *EMPTY*, the auxiliary group key for every member in T$_{Main}$ (M$_1$, M$_2$ or M$_3$) stays the same.

Leave Protocol

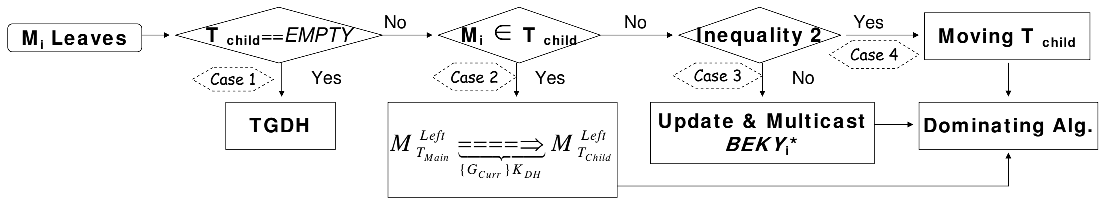

Figure 4.7: TGDH+: Leave Protocol

In this subsection, the following two issues need to be addressed.

- *How to maintain the key tree structure when one group member leaves.*

- *How to update the group key.*

Figure 4.7 shows the outline for the TGDH+ leave protocol which is the same as that for TGDH-H except for *case 2*. For all cases, the strategy of key tree structure maintenance is the same as that for TGDH-H. When case 1, 3 or 4 occurs, the leave protocol for TGDH+ releases all the group member auxiliary keys and the rest is the same as that for TGDH-H.

If *case 2* occurs in which T_{child} is available and $< h, y >$ is within T_{child}, unlike TGDH-H which utilizes the dominating algorithm, the leave protocol for TGDH+ is different.

As shown in Figure 4.8, to obtain performance gain, this leave protocol does not update the DH-based keys in the key tree for case 2 but updates the group key via *Hash* with the auxiliary group key as input. The specific idea behind this proposal is that group members in T_{main} can be aware of key material which is not known by members in T_{child}. Therefore, after a member which belongs to T_{child}, leaves, the group members in T_{main} can calculate a new group key which cannot be compromised by the group members in T_{child}, including the leaving one. Then, a designated member in T_{main} delivers the new group key to a designated member in T_{child} within a secure channel, who, in turn, sends the group key to other members in T_{child} via a secure multicast channel.

The following is a method for calculating the current group key, G$_{current}$, and for updating the auxiliary group key G$_a$.

Group Key Updates: When group member M$_{n+k}$ $\in \{M_{n+1} \ldots$ M$_{n+j}\}$ where $0 < k \leq j$ leaves, the leftmost shallowest leaf node called the sponsor in T_{main} generates a

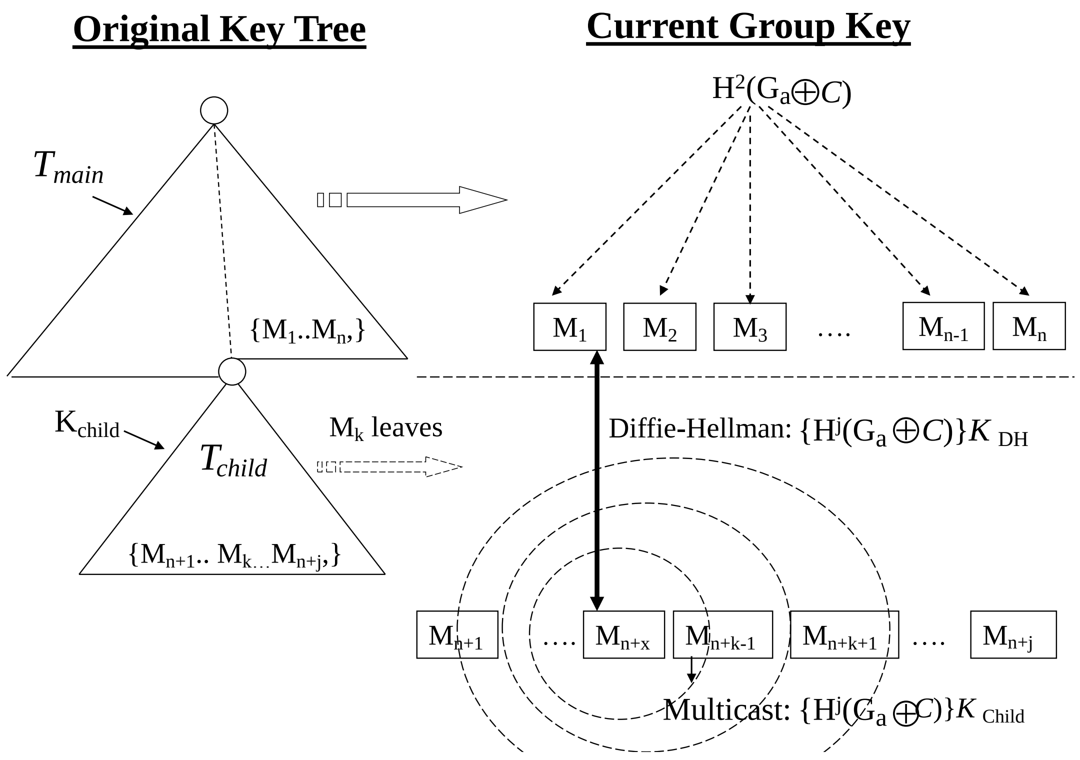

Figure 4.8: TGDH+: Group Key Updates for Case 2

new random value C. The sponsor encrypts C with G$_a$ and multicasts the ciphertext to all group members in T_{main} . Consequently, every group member $\in$ {M$_1$ ... M$_n$} can calculate a new group key G$_{current}$ = H^j(G$_a$ $\oplus C$). Next, the sponsor will establish a secure channel with the sponsor of the child key tree to deliver the current group key. As shown in Fig 4.8, the leftmost shallowest leaf of T$_{main}$, for example, M$_1$ launches a 2-party DH scheme with the leftmost shallowest leaf of T_{child}, for example, M$_{n+x}$, to generate a shared key, which is used to encrypt G$_{current}$ = H^j($G_a \oplus C$). After using the dominating algorithm to update the keys associated with the nodes in T_{child}, M$_{n+x}$ multicasts BKEY$_{n+x}$ || (G$_{current}$)K$_{child}$ where K$_{child}$ is the new sub group key associated with the root of T_{child}. Therefore, every group member in T_{child} can calculate the new sub-group key and decrypt G$_{current}$. Notice that G$_{current}$ =

$\mathrm{H}^{j-x}(\mathrm{G}_a \oplus C)$ when the (x+1)th group member leaves and all leaving x+1 members belong to the child key tree. For example, when the second group member in T_{child} leaves, $\mathrm{G}_{current} = \mathrm{H}^{j-1}(\mathrm{G}_a \oplus C)$. When the third group member in T_{child} leaves, $\mathrm{G}_{current} = \mathrm{H}^{j-2}(\mathrm{G}_a \oplus C)$.

Auxiliary Group Key Updates: If there is no new group member joins, there is no need to update C. However, if a new group member joins which is inserted into T_{child}, C should be re-generated and its ciphertext encrypted by G_a will be delivered to all other group members in T_{main}.

4.3 Contribution # 2: Periodic Group Rekeying Schemes

Although proposals such as TGDH-H and TGDH+ are more efficient than previous popular solutions such as TGDH or STR, challenges still exit when the rate of join/leave requests is too high. They introduce inefficiency and *out-of-sync* problems [68] and cannot process rekey requests in time, especially for resource-limited networks. An alternative way is to develop a time-driven group rekey scheme, namely, a periodic batch rekey which can be utilized to replace the member-driven group rekey scheme.

Periodic rekey schemes can alleviate the out-of-sync problem and improve efficiency. Within this technique, all join and leave requests are processed in a batch at the end of each rekeying interval. This means that the group members who need to leave can stay longer and new group members have to join later. Periodic batch rekeying is a trade-off between group key security and performance.

This research proposes an efficient periodic contributory group batch rekey scheme which is different from previous proposals. It handles join requests individually and deals with leaving requests in a periodic batch manner. The focus is to minimize the computational cost and communication overhead required. The rate of join/leave requests is not an intractable problem for the group key scheme if the periodic batch rekey strategy is utilized.

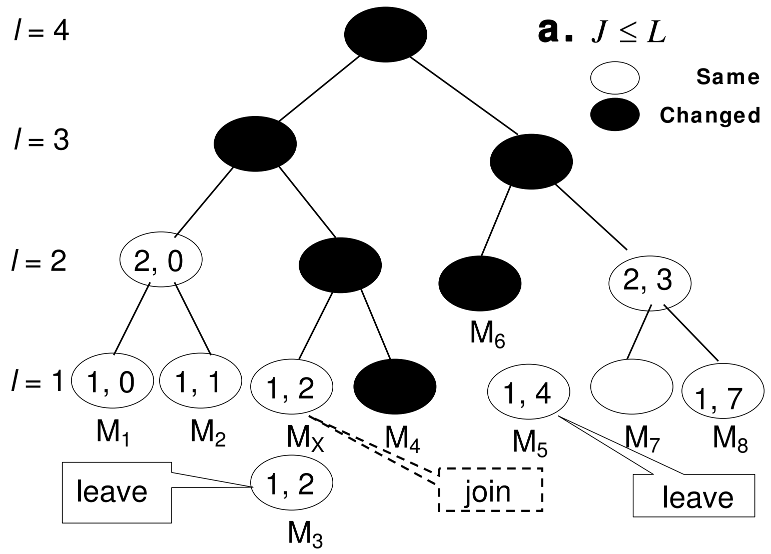

Figure 4.9: Periodical Group Key - Fewer Join Members than Leave Members

However, the periodic batch rekey strategy leads to a *vulnerability window* [68] which is the period of time starting at the first joining or leaving request for the rekey interval, and ending at the end of the rekey interval. If the vulnerability window is too long, security can be compromised. However, it can alleviate the out-of-sync problem and improve efficiency. Thus, the periodic batch rekey strategy is a trade-off between group key security and efficiency. The periodic batch rekey strategy is used in entertainment/education applications which requires a security service which is not very

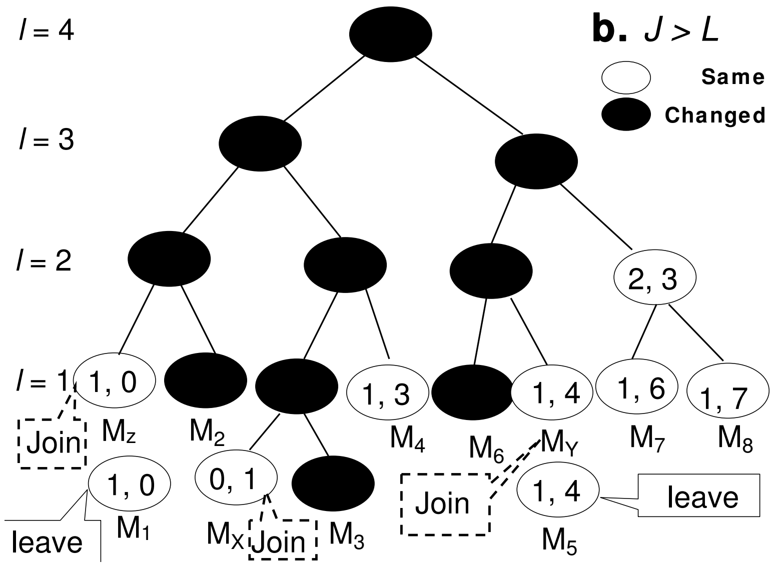

Figure 4.10: Periodical Group Key - More Join Members than Leave Members

stringent.

4.3.1 Join Protocol

The following two issues need to be addressed for this join protocol:

- *How to insert a new leaf representing a new group member into a key tree?*

- *How to update the group key?*

Insertion strategy for group members joining

The group membership events below are the same as that described in TGDH-H.

$$M_x^{Leave}, M_{n+1}^{Join}, M_{n+2}^{Join}, ..., M_{n+j}^{Join}, M_y^{Leave} | where \quad j \geq 0$$

The leave request of M_x/M_y, may or may not happen;

the join request of $M_{n+1}/\ldots/M_{n+j}$ may or may not occur.

In this rekeying interval, our join protocol's policy first replaces the new join group member with openings in the key tree resulting from leaving members. By contrast, if there are no such openings, new group members such as $\mathrm{M}_{n+2}, \ldots \mathrm{M}_{n+j}$ are appended in one child key tree, namely, T_{child}.

According to this, when group member M_{n+1} requests to join, the leaf node representing leaving member M_x is replaced with that of M_{n+1}. When group member M_{n+2}, requests to join, there is no opening caused by any leaving members and meanwhile, T_{child} is not available either. Therefore, the proposed strategy will generate T_{child} with the root at insertion point T_{Main}. (*refer to Fig. 4.2 (a) and (b)*). The following join request, $\mathrm{M}_{n+3}, \ldots \mathrm{M}_{n+j}$ can be appended into T_{child} at T_{child}'s insertion point (*refer to Fig 4.2 (a)and (c)*). The insertion points for T_{child} and T_{Main} are the shallowest leftmost nodes of T_{child} and T_{Main}, respectively.

Group key updates

Once a group member join request is sent from M_i', every group member updates the current group key G with H(G). If there is an opening caused by a leaving member, it is replaced with the new member. The sibling of M_i' (if it is a leaf node) or the leftmost offspring leaf for the sibling of M_i' works as the sponsor, S. Otherwise, following the strategy of TGDH, the shallowest leftmost leaf node is selected as a sponsor, S. S and M_i' initiate a two-party DH key exchange scheme to generate the secret key, K.

S sends M_i' the ciphertext $\{H(G)\}_K$. K also works as the key for S and M_i's parent node.

For example, Fig. 4.2 (b) is the original tree structure in which the insertion point for T_{main} is located at node $< 1,1 >$. In Fig. 4.2 (c), M_4 joins and a new leaf $< 2,2 >$ is generated to represent M_4. Since there is no opening caused by a leaving member, the old $< 1,1 >$, representing M_3, is renamed $< 2,3 >$, which works as the sibling of $< 2,2 >$. And well, a new intermediate node $< 1,1 >$ is generated to work as $< 2,3 >$ and $< 2,2 >$'s parent. $K_{<1,1>}$ is calculated with the launching of a two-party DH between M_4 and M_3, which is selected as the sponsor. M_3 sends $BKs^*@SP_3+BK'_{<2,3>}+\{H(G)\}K_{<1,1>}$ to M_4. M_4 calculates $K_{<1,1>}$ and decrypts the ciphertext H(G). In Fig. 4.2 (d), M_5 joins and it is appended into T_{child} rooted with $< 2,2 >$. The rest of the procedure is done in a similar manner.

Notice that the mutual authentication between the sponsor and the new group member will deploy previous mature solutions such as the ID-based authentication described in section 3.1.4.

4.3.2 Rekeying Protocol at the End of the Interval

For a group G with n users, $\{M_1, M_2 \ldots, M_n\}$, in every rekeying interval, consider a *J Join & L Leave* scenario (J members need to join and L members need to leave the group). At the end of the rekeying interval, the proposed scheme follows the method described below:

If $J \leq L$, the key tree is updated as follows: the leftmost leaving nodes should be updated with the new members with priority. J of the L departed nodes are replaced with the J join nodes following the one-to-one map, $\{M_i^{Depart} \rightarrow M_i^{Join}\}$. The remaining $L - J$ departed nodes' siblings will be promoted to their parents' positions and their parents will be erased. So, the ***PVs*** and ***PBs*** of L key paths should be updated. Fig. 4.9 demonstrates this procedure in which M$_x$ joins, while M$_3$ and M$_5$ leave. M$_x$ takes M$_3$'s position while M$_3$ leaves and M$_6$ is promoted to its parents' position because M$_5$ leaves. ***PVs***@M$_4$ and ***PVs***@M$_6$ are updated by the sponsors M$_4$ and M$_6$ respectively at the end of the interval.

New members may join before any members leave. Therefore, if there is no available opening, this method allocates the new members in the child key tree first and then relocate them to openings at the end of the interval.

If $J > L$, T$_{child}$ should be removed to replace the leftmost shallowest leaving leaf. Then, $L-1$ new group members, take the places of the $L-1$ departed members. The remaining $J - L + 1$ new group members stay in T$_{child}$. So, the ***PVs*** and ***PBs*** of L key paths and all the nodes in T$_{child}$ should be updated. The ASAP protocol, which will be introduced later, is used to handle this *J Join & L leave* case. For example, Fig. 4.10 demonstrates this procedure in which M$_x$, M$_y$, and M$_z$ join, with M$_1$ and M$_5$ leaving. M$_y$ and M$_z$ take M$_5$ and M$_1$'s positions respectively. M$_x$ plays the role of M$_3$'s sibling after M$_x$ joins. Keys associated with M$_2$, M$_3$ and M$_6$ are updated by the sponsors M$_2$, M$_3$ and M$_6$, respectively.

Amortized Signing & Amortized Path (ASAP) protocols

In this section, a method will be introduced for updating the key tree in an authenticated manner. First, the Amortized Path (AP) protocol based upon *Dominating Key Path Algorithm* is presented. Then, the Amortized Signing (AS) algorithm to authenticate multicast messages is proposed.

Amortized Path Protocol

In a key tree, while k key paths should be updated, every sponsor utilizes *Dominating Key Path* to decide its dominating key path and the key paths it dominates. Then, every pair of the dominating sponsor and the dominated sponsor launches an ID-based two-party Diffie-Hellman key exchange scheme [26] to generate the new secret key, K', between them. The dominated sponsor encrypts the updated public keys on its key path and other public keys it collects from the sponsors it dominates with the key K' and then delivers the ciphertext to the dominating sponsor. This is repeated until the leftmost sponsor receives all the updated public keys of k key paths, signs them with the Amortized Signing algorithm (AS) and multicasts the updated public keys on the key tree. In contrast to TGDH and Queue-Batch, the proposed AP protocol utilizes unicasts to replace multicasts and uses encryption operations to replace signing operations.

For example, in Fig. 4.10:

1st round: The key path for M_3 is dominated by that of M_2. M_2 sends M_3: $PB_{<2,0>}$

M_3 sends M_2: $PB_{<2,1>}+\{PB_{<0,2>}+PB_{<1,2>}\}\ PV_{<3,0>}$

2nd round:

M_2 sends M_6: $PB_{<3,0>}$

M_6 sends M_2: $PB_{<3,1>}$+{ $PB_{<1,5>}$+ $PB_{<2,2>}$} $PV_{<4,0>}$

3^{rd} round:

M_2 signs and multicasts $PB_{<1,1>}$+ $PB_{<2,0>}$+ $PB_{<3,0>}$ +$PB_{<0,2>}$+ $PB_{<1,2>}$+ $PB_{<2,1>}$ + $PB_{<1,5>}$+ $PB_{<2,2>}$ + $PB_{<3,1>}$

All group members update the PBs and PVs on their key path and obtain the group key.

Table 4.3: Amortized Path (AP) Protocol

Protocol 4: *Amortized Path Protocol (AP)*
Every Sponsor M$_i$: 1: updates its key path until it cannot 2: launches a 2-party Diffie-Hellman with the sponsors it has dominated or dominates 3: **IF** there are new keys which can be updated for M$_i$, 4: go to step 1 5: **else** 6: encrypts all updated public keys it knows and sends ciphertext to its dominating sponsor
Leftmost Sponsor M$_L$: 1: Signing all updated public keys on the key tree via *AS* 2: multicasts the result
All group members: 1: after receiving the *PBs* from leftmost sponsors M$_L$, updates the *PVs* and group key on its key path.

Authenticated Unicast and Multicase

ID-based DH providing authentication functionality is utilized in a join protocol to generate the shared key between the new member and the sponsor. It is also used in the AP protocol between the *dominating sponsor* and the *dominated sponsor*. Specifically, after a shared key is generated, the group key or public keys should be encrypted and delivered from one party to another. Specifically, the unicast message is encrypted with symmetric encryption algorithms. To protect this ciphertext against insider attacks, efficient hash-based message authentication codes (MAC) known as HMACs [12] are used to authenticate the unicast transmission

According to the requirements of the group rekeying schemes, rekeying multicast messages should be authenticated and reliable. Unlike unicast schemes, a multicast authentication scheme based solely upon MAC cannot be both efficient and collusion resistant [6]. Consequently, to guarantee a reliable multicast service for authentication information, Park and Siegel [23] propose the Signature Amortization Information Dispersal Algorithm (SAIDA) to encode authentication information with Rabin's Information Dispersal Algorithm (IDA) [27].

This proposal signs multicast messages with an amortized signing technique, namely M-SAIDA (Modified Signature Amortization Information Dispersal Algorithm) in which an RSA digital signing algorithm signs the hash of the entire multicast rekeying messages. This is a previously proposed modification of SAIDA [9]. In addition to the efficient signing feature, some techniques for guaranteeing the reliability of not only the digital signatures but also the rekeying messages is proposed. The proposed algorithm (M-SAIDA) outlined in Table 4.4, satisfies the requirements of both zero packet loss and computational efficiency. The implementation of the IDA which presents reliable transmission for data packets by introducing some amount of information redundancy have been documented well [23, 27]. IDA splits the source data, for example, PB_i, into n pieces, which are then encoded by the IDA algorithm. At the receiving end, the IDA can reconstruct PB_i after receiving any m pieces where $m < n$. However, guaranteeing zero packet loss comes at the cost of increased communication overhead.

To update the key tree, this proposal multicasts rekeying messages once according to *ASAP*. Hence, one RSA signing is required. In TGDH, every sponsor performs the signing [*1, log*$_2$*k*] times. Therefore, in total, all sponsors sign $2k-1$ times for TGDH, where k is the number of sponsors. In *Queue-Batch*, even using the *AS* algorithm, $J+L-1$ signings are required.

4.3.3 Merging and partition protocols

When the group is divided into sub-groups, such as during network disruptions, the partitioning protocol will treat the members who cannot be in contact with the group as leaving members. In this case, each group member will handle the *0 join & L leave* scenario. In a similar way, when sub-groups merge due to network re-connections, the merging protocol deals with the *J join & 0 leave* scenario. For every sub-group, the group member hosting the leftmost shallowest key path is treated as the sponsor for the sub-group which generates the new session secret key, updates keys on its key path and multicasts the updated keys.

The rekeying, merge and partition protocols can use the *ASAP* protocol to handle the *J join & L Leave, J join & 0 Leave* or *0 Join & L leave* scenarios, respectively.

Table 4.4: M-SAIDA Algorithm

Algorithm 2: M-SAIDA

Sponsor M_i :

INPUT : Public keys PB_1, $PB_{2...}$ PB_n;
OUTPUT: Public keys and signature encoded by algorithm IDA-Encode [27];

```
Notation. || : concatenation; H = NULL;
/ * the hash value for $PB_1$, $PB_{2...}$ $PB_n$ */
for $1 \leq i \leq n; i++$;
    H = Hash(H||$PB_i$);
    /* Hash is an secure one-way-hash function such as SHA-256*/
end for
$M_i$ multicast (IDA-Encode ($PB_1$), ...IDA-Encode($PB_n$), IDA-
Encode ($Sign_{RSA}$(H)));
/*RSA is used because it is efficient in verification.*/
```

The Receiver $M_1 \ldots M_n$:

INPUT : Public keys and signature encoded by algorithm IDA-Encode [27];
OUTPUT : Public keys and signature;

```
for $1 \leq i \leq n; i++$;
    IDA-Decode ($PB_i$);
    H = Hash(H||$PB_i$); /* Hash is an secure one-way-hash function such
    as SHA-256*/
end for
$Verify_{RSA}$ (H, IDA-Decode ($Sign_{RSA}$(H)))
```

4.4 Contribution # 3: Maximum Matching Algorithms

Many scenarios demonstrate that the group key is established among a number of members starting at a certain time. For example, when an event is going to start at a certain time, several group members co-operate with each other to generate the group key at this time. Another example would be, after the group communication is partitioned, a group is divided into two or more sub-groups. Sub-groups with fewer members should generate the new group key among sub-group members at a certain time. Both cases can take advantage of the algorithms presented below. contribution focusses on the efficiency of a binary tree-based group key establishment. To reduce its communication overhead, Maximum Matching algorithms (M2) are proposed to place the group members into pairs and then the blinded keys can be exchanged in every node pair or node set pair to accomplish the binary tree-based group key agreement scheme. While most previous algorithms [4], [62] have been primarily sequential, M2 is a parallel algorithm. Furthermore, M2 algorithms require that each member keeps track of only its matched partner instead of remembering the entire group member list which reduces the storage cost for every node. Therefore, the M2 algorithms reduce communication costs and decrease the number of rounds during the group key generation procedure.

4.4.1 Maximum Matching Algorithms

To deliver blinded keys associated with nodes in the key tree, a regular solution is to multicast/broadcast them. Unfortunately, this method requires large communication bandwidth so it is not an efficient choice, especially for MANETs, the resource-limited network settings. To lessen communication overhead, in every level of the key tree, every node requires only its sibling's blinded keys to continue the Diffie-Hellman operations. First, two nodes or node sets are matched into a pair and then the partial keys (blinded keys) between the node pairs or node set pairs are exchanged via unicast to decrease communication costs. This technique should be used at each group key establishment round until the node key of the key tree's root is calculated.

Two Maximum Matching Algorithms

A matching on a graph is a set of edges, no two of which share a vertex. A maximum matching, M, contains the greatest number of edges possible. M is a maximum matching if no matching on the graph contains more edges than M. Edmond [16] proposed the *blossom algorithm* to find the maximum match in a graph. The matching algorithm [83] is the latest for random graphs and the self-stabilizing Synchronous Maximal Matching (SM2) algorithm [19] is the latest proposal for ad hoc networks. This thesis utilizes/proposes two solutions, one is SM2 and the other is Depth-first Spanning Maximal Matching (DST-M2), which will be described later.

Synchronous Maximal Matching (SM2) Algorithm

The Synchronous Maximal Matching (SM2) algorithm assumes that each node n_i maintains the identities of its neighbours in an array *neighbors*(n_i) by sending and listening to the beacon (keep-alive) message periodically. It is fault-tolerant in the sense that it can detect occasional new link creations in the network. SM2 demands

O(n) steps for a general graph where n is the group size. In this algorithm group members can make proposals to their neighbours or remote members to match with them. If the proposal is accepted, the two members are matched and they do not accept other proposals or make proposals to others. If there is no response to the proposal, the member should withdraw it and make a new one to other members. In addition to the three original rules proposed in the SM2 algorithm, the fourth rule for handling remote nodes is described at Table 4.4:

Depth-first Spanning-Tree Maximum Match (DST-M2) Algorithm

The other M2 algorithm, based upon the Depth-First Spanning Tree [82] and [21], is introduced in this proposal as well. The advantage of DST-M2 is that fewer rounds, O($\log_2 n$) are required to reach maximum matching where n is the group size. However, to take advantage of this strength, the topology of the network is assumed to be known by every group member in advance and this can be achieved with such technologies as Global Positioning System (GPS) location systems. Furthermore, to generate the initial tree, every node requires O(n) compuational cost but unlike the shortest path algorithm, no communication overhead is required.

4.4.2 Vertex Shrinking Algorithm

This algorithm is proposed to combine matched nodes into a node set which can be treated as one node set for the next round (level).

Table 4.5: Synchronous Maximal Matching (SM2) Protocol

Protocol 5: Synchronous Maximal Matching (SM2) /* MATE (a) = b means that a's matched partner is b. We say that n_i and n_j are matched if MATE (n_i) = n_j & MATE (n_j) = n_i. where n_j and n_i, are group members */	
Rule 1:	/* n_i accepts the proposal from n_j. $*$ / **If** MATE (n_i) = *NULL* and n_j sends a request to n_i, **Then** MATE(n_i) = n_j .
Rule 2:	/* n_i makes the proposal to n_j:*/ **If** MATE (n_i) = *NULL* and n_j is with the smallest ID among un-matched group members. **Then** n_i sends a request to match n_j, and MATE(n_i) = $n_{j\_}$
Rule 3:	/* n_i withdraws the proposal */ **If** MATE (n_i) = n_j but MATE (n_j) $\neq n_i$ **Then** MATE (n_i) = *NULL*
Rule 4:	/* Isolated group members make proposal to remote un-matched group members*/ **If** MATE (n_i) = *NULL* and ($\forall n_j$, MATE (n_j) $\neq n_i$) **Then** n_i sends requests which are replied to by the nearest non-matched group member.

Table 4.6: Depth-first Spanning Tree algorithm M2 (DST-M2) Protocol

Algorithm 3 - Depth-first Spanning Tree algorithm M2 (DST-M2) /*V_i is a node or a node set.*/	
Step 1:	All group member nodes launch minimum Depth-first Spanning Tree algorithm (DST). The roots will be the nodes with the smallest ID compared with their neighbours. We get a collection of independent spanning trees.
Step 2:	DST is executed to get a full spanning forest $\boldsymbol{F}$ which connects all spanning trees
Step 3:	The post-order walking algorithm over spanning forest $\boldsymbol{F}$ is executed by the source node. Then, a sequence S = (V_1, V_2, . . . V_n) is generated
Step 4:	Delete all repeated V_i in S. If V_i is not a group member node, it should be deleted in S, too. A new sequence S' = (V_1', V_2', . . . V_n') is obtained.
Step 5:	For S' = (V_1', V_2', . . . V_n'), $V_{2i'-1}$ and $V_{2i'}$ are matched *where* $1 \leq i \leq n/2$

Table 4.7: Vertex Shrinking Algorithm

Algorithm 4: Vertex Shrinking Algorithm (VS) /* Vertex Shrinking algorithm combines every couple (Set1, Set2), the outputs of M2 algorithm.*/	
Step 1:	Initializes a new node set *newset* = *NULL*
Step 2:	If node $n_i \in$ *Set1* or *Set2*, **Then** $n_i \in$ *newest*
Step 3:	$\forall e = (n_i, n_j) \in \mathbf{E}$ where e is an edge and **E** is a set including all edges. **If** both n_i and $n_j \in$ *Set1* or *Set2,* **Then** e is marked as an inner line $\in \mathbf{L}$.

4.4.3 Utilization of Maximum Matching Algorithms

In every round of the group key establishment or every level of the binary key tree, M2 and VS algorithms are used together to fulfill the task.

Let $\mathbf{M} = (\{\mathbf{m}_1; \ .\ \ .\ \ .; \ \mathbf{m}_n\}; \ \mathbf{L})$ be the topology of a multi-hop location-aided wireless ad hoc network where $\mathbf{m}_i$ is a node and $\mathbf{L}$ is the set of communication channels established by two nodes. So, $\mathbf{M}$ can be modelled as an undirected graph $\mathbf{G} = (\{\mathbf{n}_1; \ \mathbf{n}_2; \ .\ \ .\ \ .\ \ ; \ \mathbf{n}_n\}, \ \mathbf{E})$ where vertex $\mathbf{n}_i$ corresponds to node $\mathbf{m}_i$ in $\mathbf{M}$ and $\mathbf{E}$ denotes the set of edges in $\mathbf{L}$. There is an edge in $\mathbf{E}$ between a pair of vertices $\mathbf{n}_i$ and $\mathbf{n}_j$ if nodes $\mathbf{m}_i$ and $\mathbf{m}_j$ in $\mathbf{M}$ enable successful communication directly.

Table 4.8: Integrated Algorithms to Achieve Group Key Establishment

Algorithm 5: Integrated Algorithms /* this algorithm is developed to construct the key tree and generate the group key */ *Round* l ($1 \leq l \leq$h-1)	
Step 1:	$M2$ algorithm provides a set of triples (l, n$_i$, n$_j$)$\ldots$
Step 2:	Node or node set n$_i$ and n$_j$ simultaneously exchange intermediate blinded keys with each other. All group members compute node keys and blinded keys associated with the node at the $(l+1)^{th}$ level on its *key-path* according to *TGDH*.
Step 3:	*Vertex Shrinking* (VS) algorithm combines every matched couple into one node set.
Step 4:	**If** the new node set contains all group member nodes, **Then** exit **Else** l ++, goto step 1.

According to this algorithm, we find that at the l^{th} round (level) where $1 \leq l \leq h-1$, the M2 algorithm matches pairs of vertices, for example, (n_i, n_j), in graph G. Based upon triples (l, n_i, n_j), nodes or node sets m_i and m_j exchange their intermediate blinded keys with each other at level l in the key tree. The vertex shrinking algorithm updates graph G by combining every pair (n_i, n_j) into a new super node set, namely, (n_in_j). The procedure is repeated until only one node set, which contains all group member nodes, is left in graph G. Figure 4.11 - 4.13 and Figure 4.14 - 4.16 demonstrate step-by-step procedures for running an Integrated Algorithm (Table 4.8) by using the SM2 and DST-M2 algorithms. According to every round's triple outputs shown in Figure 4.11 - 4.13 and Figure 4.14 - 4.16, the procedure for constructing every level of the key tree **T** is shown in Figure 4.17 - 4.19.

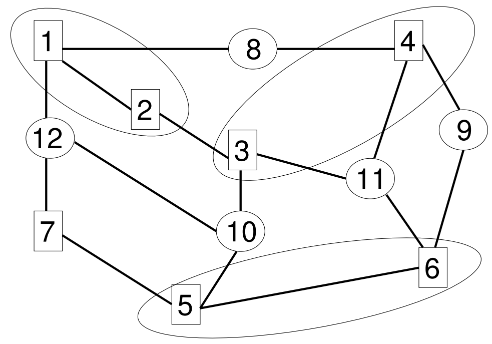

Figure 4.11: Intermediate Result by Using the SM2 Algorithm - Round 2

Figure 4.11 shows the topology of the network upon which the Integrated Algorithm (Table 4.8) is processed to output the triples (l, n_i, n_j) round by round. If the Integrated Algorithm (Table 4.8) uses the SM2 algorithm, every group member can obtain the triple results itself. For instance, group member M_1 can process the sorted triple lists, {2, M_1, M_2}, {3, M_1M_2, M_3M_4}, and {4, M_1M_2 M_3M_4, $M_5M_6M_7$}.

Result for Figure 4.11 Match Pairs:(1, 2) (5, 6) (3, 4) triple results: (2, $\underline{1, 2}$)(2, $\underline{3, 4}$)(2, $\underline{5, 6}$)

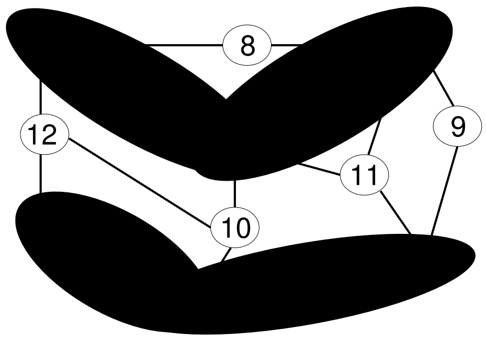

Figure 4.12: Intermediate Result by Using the SM2 Algorithm - Round 3

Result for Figure 4.12 Match Pairs:(12, 34) (56, 7) triple results: (3, $\underline{12, 34}$) (3, $\underline{56, 7}$)

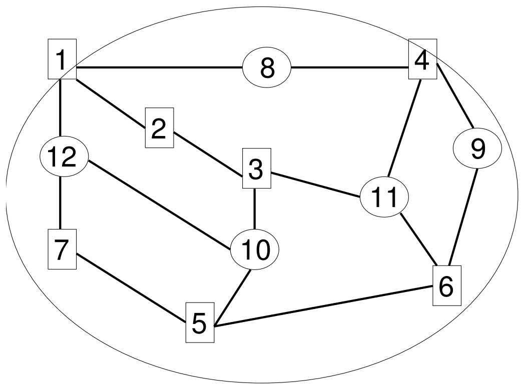

Figure 4.13: Intermediate Result by Using the SM2 Algorithm - Round 4

Result for Figure 4.13 Match Pairs:(1234, 567) triple results: (4, $\underline{1234}$, $\underline{567}$)

Result for Figure 4.14 Post-order walk-list: (1,2,3,4,5,6,7) Match Pairs:(1, 2) (5, 6) (3, 4) triple results: (2, $\underline{1, 2}$)(2, $\underline{3, 4}$)(2, $\underline{6, 5}$)

Result for Figure 4.15 Post-order walk-list: (12,34,65,7) Match Pairs:(12,34) (56,7) triple results: (3, $\underline{12}$, $\underline{34}$)(3, $\underline{65}$, $\underline{7}$)

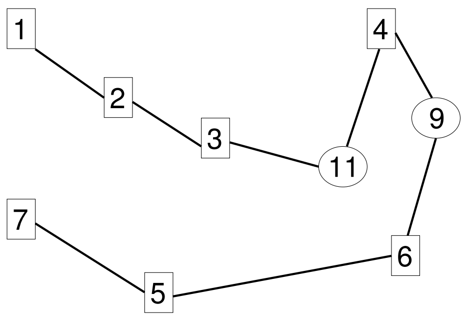

Figure 4.14: Intermediate Result by Using the DST-M2 Algorithm - Round 1

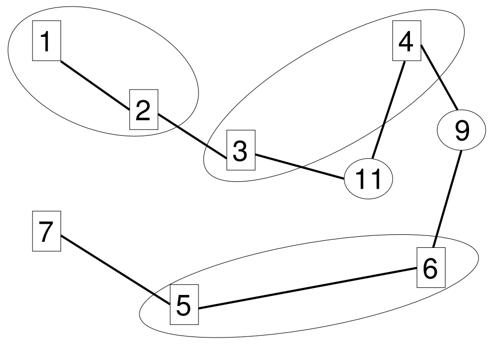

Figure 4.15: Intermediate Result by Using the DST-M2 Algorithm - Round 2

Result for Figure 4.16 Post-order walk-list: (1234,657) Match Pairs:(1234) (567) triple results: (4, $\underline{1234}$, $\underline{567}$)

If the Integrated Algorithm (Table 4.8) uses DST-M2, it is assume that every node can be aware of the topology of the network by using a GPS system. It outputs a series of triple results (l, n_i, n_j) and forwards them to the node or node set n_i and n_j with the support of the *Prufer code* algorithm [14]. For instance, group member M_2 is offered the sorted triple list, {2, M_1, M_2}, {3, M_1M_2, M_3M_4}, and {4, M_1M_2 M_3M_4, $M_5M_6M_7$}.

Consequently, based upon these triples, nodes exchange blinded keys with matched

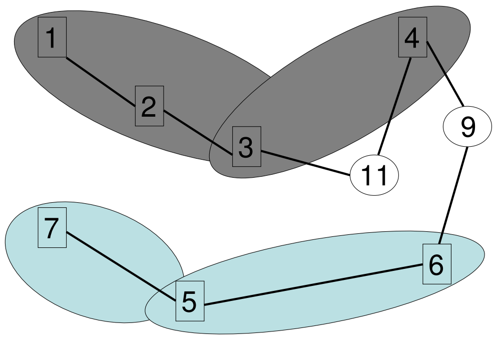

Figure 4.16: Intermediate Result by Using the DST-M2 Algorithm - Round 3

partners in every round and the height-balanced binary tree can be constructed. As shown in Figure 4.17 - 4.19, a binary tree-based group key establishment scheme such as TGDH can be accomplished and the group key is calculated by every group member. Notice that while one group member node is out of service, the maximum matching algorithms can treat the node as a member who is leaving or is not available. Consequently, the failure node's partner need not do any processing and simply waits for the next round.

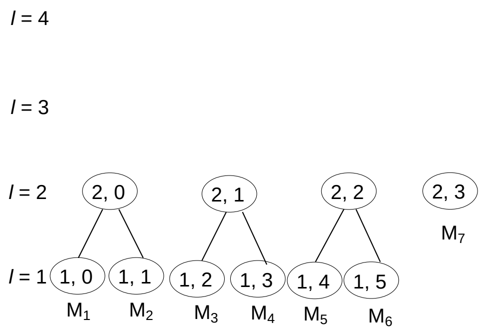

Figure 4.17: Key Tree Construction with the M2 Algorithm - Round 1

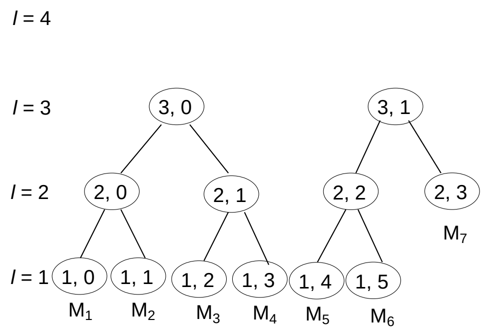

Figure 4.18: Key Tree Construction with the M2 Algorithm - Round 2

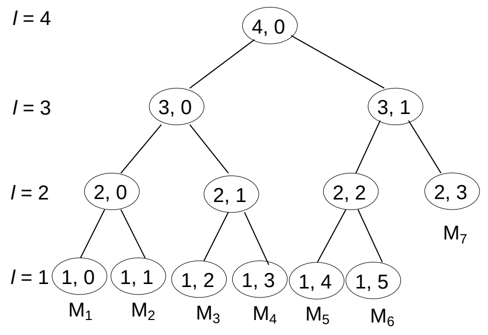

Figure 4.19: Key Tree Construction with the M2 Algorithm - Round 3

4.5 Contribution # 4: Hybrid Architecture

4.5.1 Overview of the Fourth Contribution

For some cases which occur in hostile environments, the group key management scheme should switch from centralized to contributory schemes. One example from a military scenario follows. A collection of wireless mobile devices carried by soldiers or a tank cooperates in relaying packets. In such a scenario, mobile nodes establish routes dynamically among themselves to form networks. However, all nodes except for the one with the tank, have limited battery power and processing capacities. For the sake of power-consumption and computational efficiency, the tank can work as the key server while a centralized group key management scheme is deployed. Nevertheless, it is possible for the tank to be out of service during adverse conditions. Alternatively, some of the soldiers may not be able to be in contact with the tank. In case of such emergencies, a straightforward solution is to launch a contributory group key agreement, for example, TGDH [25], TGDH-H, or TGDH+, among the nodes which cannot be in contact with the centralized key server. However, the disadvantages are: 1) every node should install two sets of group key management schemes, 2) every node should store two sets of individual keys, auxiliary keys and group keys, 3) it uses up critical time to generate the new contributory group key, and 4) generating the new group key requires additional computational cost and communication overhead.

In this sub-section a novel hybrid group key management protocol which removes the above drawbacks is proposed. In particular, this protocol enables fast switching between centralized and contributory schemes with minimal communication costs and computational overheads while using fewer keys. Specifically, the hybrid architecture has centralized and contributory group key management schemes using a Logical Key Hierarchy (LKH)-based management technique to maintain one set of keys in a binary key tree. When the key server is online, the computationally efficient centralized management is deployed. While the key server is out of service for all group members or part of the group members and for handling network disconnections, an efficient contributory scheme, such as TGDH-H or TGDH+ is used to update the group key shared by the group members who cannot connect with the key server.

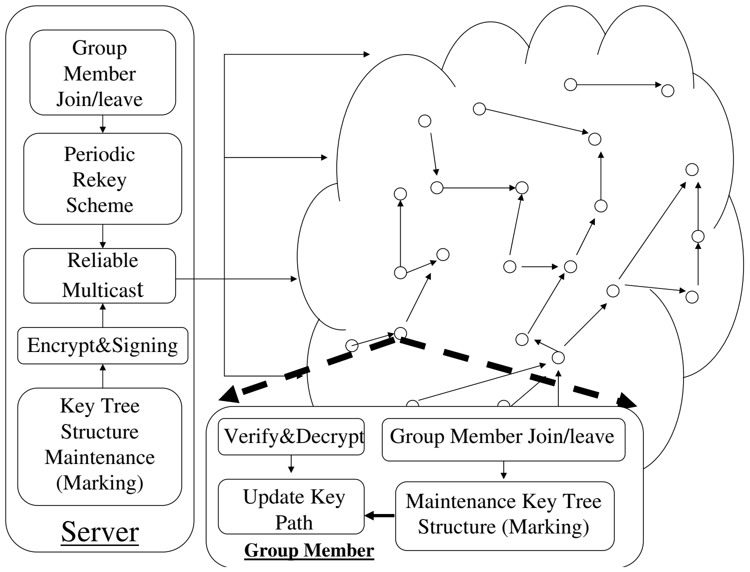

Figure 4.20: Centralized Group Key Management

4.5.2 Hybrid Architecture

This section will present the hybrid architecture which combines the advantages of the centralized approach's efficiency and the contributory scheme's fault tolerance. The basic idea behind the hybrid architecture is that when the key server is down (off-line), then group key management is done using a contributory scheme such as TGDH-H or TGDH+. If the key server is on-line, then there are two possibilities. If all the group members are able to access the key server (no partitioning of the group), then a centralized scheme such as LKH+ [61] is used in which the key server is responsible for calculating and delivering the intermediate keys associated with the binary key tree following formula (3.5), since the key server is deemed to have a high processing capability. On the other hand, if the group is partitioned (some of the members are not able to access the key server), then a combination of the two schemes is used – the members with access to the key server use the centralized

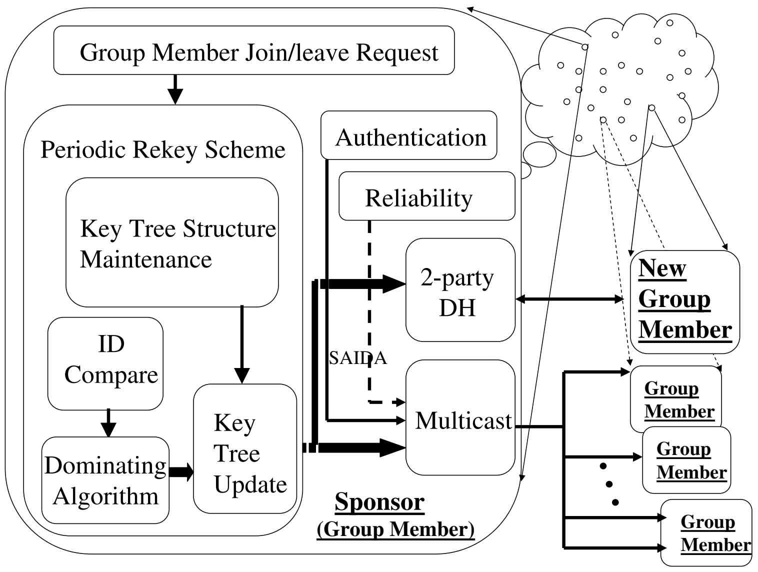

Figure 4.21: Contributory Group Key Management

scheme while the others use the contributory scheme. Both of them follow the TGDH key tree to update the node keys and blinded keys associated with the nodes on the binary key tree.

Figures 4.20 and Figure 4.21 demonstrate the required components for implementing the centralized and the contributory group key management schemes, respectively. As mentioned earlier, it is assumed that every group member or the key server is aware of any group membership changes based upon the *group member join/leave request* component which is part of the *View Synchrony* service. Furthermore, the server and the members modify the key tree structure according to the *key tree structure maintenance* component which follows that of TGDH or binary LKH (TGDH is the binary LKH in terms of the key tree structure).

In general, centralized group key management administers the group key update for group membership changes. In Figure 4.20, the key server is responsible for updating, encrypting and authenticating rekey messages including individual keys, auxiliary keys and group keys. Furthermore, it forwards the encrypted rekey messages to other group members via the *reliable and authenticated multicast* component which is also part of the *View Synchrony* service. Every group member should utilize the *verify & decrypt* component to process the rekey messages to update its key paths and the group key.

However, the centralized scheme cannot handle situations such as key server failure or network partitions due to intermediate nodes being out-of-service or severe network congestion. This problem can be dealt with efficiently by the proposed contributory scheme. When the key server loses its availability, all group members manage the key tree themselves in a contributory mode. Since both centralized and contributory key management use the same key tree structure, no rekey operations are processed and no rekey messages are forwarded to implement the switch from the centralized scheme to the contributory scheme.

Figure 4.21 shows the components of the contributory scheme, in which the *ID-Comparison* algorithm [74] and the *Dominating* algorithm are deployed to update the key tree. The sponsor's (i.e., the group member selected by the algorithm to update and forward the keys on the key path) *reliable and authenticated multicast* component (for instance, M-SADIA) forwards to the other group members the updated public keys. Upon receipt, the other group members calculate the new group key. As in previous contributory group key management schemes [25], [26], the sponsor should process the *two-party DH key exchange* (for instance, ID-based DH) with the new group member as well.

Specifically, the LKH+ [61] scheme can be used in centralized group key schemes and TGDH-H, or TGDH+ can work as the contributory group key scheme.

4.6 Contribution # 5: Further Improvements for FEA-M

FEA-M can be deployed to encrypt group messages exchanged among group members efficiently. In this sub-section, further vulnerabilities of the FEA-M's improved variant [35] are discussed, and then improvements for overcoming those vulnerabilities are described.

4.6.1 A Vulnerability with FEA-M and an Improved Variant

A feature of FEA-M is to provide a connection between neighbouring plaintext blocks. If the adversaries replay the earlier packets, the receiver can notice the faked message. Therefore, it shows a feature which can persist against packet replay attacks. However, although it tolerates packet loss problems, Mihaljevic's improvement [35], described by formulas (3.10) and (3.11), is vulnerable to packet replay attack. For example, the adversary can obtain the earlier i^{th} ciphertext block, C$_i$'. Then, s/he captures the current i^{th} ciphertext block, C$_i$ and replaces C$_i$ with C$_i$'. In cases where **K** and **V**$_0$ are not changed, the receiver cannot be aware that the cipher text is the earlier plaintext. According to formulas (3.10), (3.11) and (4.2), what the receiver gets is P_i' rather than P_i if C_i' replaces C_i.

$$P_i' = K^{-1} \bullet (C_i' + K \bullet V \bullet K^i) \bullet K^{-(i+n)} + K \bullet V \bullet K^i \tag{4.2}$$

The reason this kind of attack works is because the improvement in [35] treats each block of plaintext independently.

4.6.2 Further Improvement for FEA-M

Compress plaintext to avoid Mihaljevic'sassumption [34]

FEA-M's security degradation due to plaintext blocks being all 0s [33], [34] can be solved with the compression algorithm. In general, before encryption by a regular encryption algorithm, the multimedia plaintext should be compressed. Figure 4.22 describes how the multimedia data is compressed, encrypted and transmitted across the insecure channel from party A to party B.

The computational complexity of this algorithm is O(n), an example of which follows. Suppose that, in the first set of plaintext blocks, there are 0s in a row with length n^2. After compression, as shown in Figure 4.23, it is no longer an all-zero block:

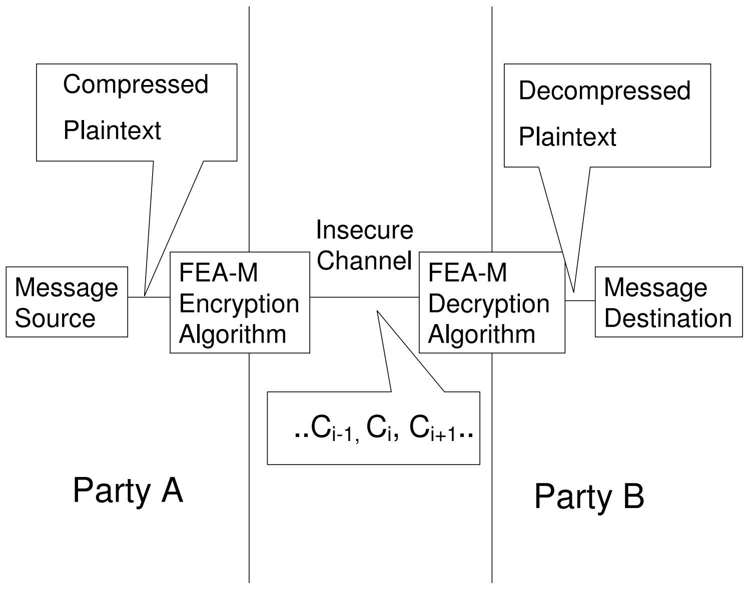

Figure 4.22: Multimedia Communication Model

The well-known Routine Length Encode (RLE) technique [32] which reduces redundant messages is deployed in a number of popular compression algorithms. In this proposal, the RLE technique is utilized to overcome the weakness of blocks being all 0s. The details of RLE are listed below.

Table 4.9: Multimedia Data Compression Algorithm - RLE

Algorithm 6: Multimedia Data Compression - RLE

INPUT : Plaintext byte stream B_1, $B_2 \ldots B_n$,
OUTPUT : Compressed Byte Stream ($*N_i$, B_i) or (B_i) where * indicates how many times the subsequent data byte repeats itself where $N_i > 1$

```
1:  B = 1st Byte ; count = 0;
2:  FOR (not end of the byte stream)
3:  While (current byte value = B)
4:  {count++; read next byte}
5:  If (count > 1)
6:  Put (*count; B);
7:  Else
8:  Put (B);
9:  B = current byte value; count = 0;
10  END FOR
```

$$\overbrace{\underbrace{......0,0,...,0}_{n^2}}^{Before}\underbrace{\Longrightarrow}_{Compress}\overbrace{......n^2,0}^{After}$$

Figure 4.23: Compression for Blocks with All 0s

4.6.3 Reliable transportation to handle packet loss

FEA-M is vulnerable to packet losses. Furthermore, the improved variant [35] is vulnerable to block replay attacks. In this sub-section, two methods for overcoming these weaknesses are proposed.

Reliable FEA-M (rFEA-M)

Table 4.10: rFEA-M Algorithm

```
Algorithm 7: rFEA-M

The Sender Party A :

INPUT : Plaintext blocks $P_1, P_2 ... P_r$,
OUTPUT : Ciphertext blocks $C_1, C_2, \ldots C_r$ , which are encoded by IDA-Encode.

1: Notation. || : concatentation; $K$: key matrix; $P_i$ : a block of plaintext
2: for $1 \leq i \leq r; i++$;
3:     $C_i$ = FEA-M-encrypt ($P_i$);
4:     A$\rightarrow$ B: IDA-Encode ($C_i$).
5: end for

The Receiver Party B :

INPUT : Ciphertext blocks, $C_1, C_2, \ldots C_r$ which are encoded by IDA-Encode;
OUTPUT: Plaintext blocks $P_1, P_2 ... P_r$, ;

1: for $1 \leq i \leq r; i++$;
2:     $C'_i$= IDA-Decode ($C_i$);
3:     $P_i$ = FEA-M-decrypt ($C'_i$);
4: end for
```

Packet loss is not a problem for applications using reliable transport protocols [e.g. the Transmission Control Protocol (TCP)]. However, in cases where applications deploying the FEA-M algorithm do not utilize such protocols, techniques which are robust against packet loss have to be used. For instance, multimedia applications such as medical imaging systems which cannot tolerate source data packet losses demand this requirement. Therefore, a robust FEA-M (rFEA-M) method for overcoming the FEA-M flaw is proposed. FEA-M is used to encrypt/decrypt data packets. To make FEA-M robust against message losses, Rabin's Information Dispersal Algorithm (IDA) [43], [41] is utilized which encodes every cipher text block, introducing some amount of redundancy. Algorithm 7 is proposed for implementing rFEA-M. Algorithm 7 provides no-packet-lost service and implements FEA-M for network settings with packet loss. However, it has higher computational costs due to the implementation of IDA to process every cipher text block. The computation complexity of IDA is $O(n^2)$ where n is the data length of plaintext.

Correction for improved variant [35] for FEA-M

To provide data source authentication against packet replay attacks for the improved variant [35], a secret-key-based message authentication code (e.g. MD5-MAC [32]) is used to process every cipher text block. Then, IDA is used to encode/decode all MD5-MAC results to guarantee that they are all received by the receiver party. This method is suitable for applications such as Internet TV or Internet Radio which can tolerate source data packet losses.

Specifically, the improved variant [35] proposed by Mihaljevic is used to encrypt/decrypt the data packets. Furthermore, to resist block replay attacks, MD5-MAC is used to process every source data block. Then, the results of MD5-MAC are encoded with IDA to guarantee that the MD5-MAC information is not lost during packet transport. The receiving party uses the IDA algorithm to reconstruct all of the MD5-MAC results it has received. Finally, the receiver can verify the integrity of every cipher block. In the following, algorithm 8 is proposed to implement this method. Algorithm 8 provides a data source authentication service for persisting against packet replay attacks for the improved variant [35]. However, it has a higher computational cost due to the implementation of IDA and MD5-MAC. The computation complexity is O(n^2) where n is the data length of MD5-MAC results and n is smaller than the plaintext size.

IDA Algorithm

Algorithm 9 focusses on the implementation of IDA which presents reliable transmission for data packets by introducing some amount of information redundancy. IDA splits the source data, for example, C_j, into n pieces, which are then encoded by the IDA algorithm. At the receiving end, IDA can reconstruct C_j after receiving any m pieces where $m < n$. However, guaranteeing zero packet loss comes at a cost of increased communication overhead. For example, for r blocks, assume every block is 4096 bits. So, n is 64, m can be 50. For Algorithm 7, *4096*r*n/m* bits of data are sent over the network and at least $4096 * r$ bits of data are received. For Algorithm 8, in addition to the source data, $64 * r * n/m$ bits of hash are sent over the network and at least $64 * r$ bits of hash are received.

According to Algorithm 9, the computation complexity of IDA is O(n^2).

Table 4.11: Correction for the improved FEA-M variant

Algorithm 8: Correction for the improved FEA-M variant

The Sender Party A :

INPUT : Plaintext blocks: $P_1, P_2 ... P_r$,where $r \geq 1$ and it is defined by applications.
OUTPUT : Ciphertext blocks, $C_1, C_2, \ldots C_r$ and H which is encoded by IDA

```
1: Notation. || : concatentation; $K$: key matrix; $P_i$ : a block of plaintext;
2: $H$: concatenated result of MD5-MAC; H is *empty*;
3: **for** $1 \leq i \leq r; i++;$
4: $H_i = MD5 - MAC(P_i, K_{upper128bits});$
5: $H = H_i \mid\mid H;$
6: $C_i$ = Improved variant in [35]-encrypt ($P_i$);
7: A→ B: $C_i$;
8: **end for**
9: A→ B: IDA-Encode ($H$);
```

The Receiver Party B :

INPUT : Ciphertext blocks, $C_1, C_2, \ldots C_r$, *and* H which is encoded by IDA;
OUTPUT : Plaintext blocks: $P_1, P_2 ... P_r$;

```
1: **for** $1 \leq i \leq r; i++;$
2: $P_i$ = Improved variant in [35]-decrypt ($C_i$);
3: **end for**
4: $H'$ = IDA-Decode ($H$);
5: **for** $1 \leq i \leq r; i++;$
6: **if** ($P_i$ is received) and ( $MD5 - MAC(P_i, K_{upper128bits}) \neq$ H'[i]);
7: *return "wrong data";*
8: **endif**
9: **end for**
```

Table 4.12: IDA Algorithm

Algorithm 9: IDA

The Sender Party A : IDA-Encode

INPUT : a block of data C_j
OUTPUT : encoded vectors $T_1, T_2 \ldots T_n$

```
(1) Split C_j into N/m pieces where N=n/8:
    $C_j = (c_1, ..., c_m), (c_{m+1}, ..., c_{2m}), ..., (c_{N-m-1}, ..., c_N) where, c_i : byte$
    $R_i = (c_{(i-1)m+1}, ..., c_{im}), where, i < N/m$
(2) Process $C_j$: following the specification of IDA, choose n vectors, let every subset of m different vectors be linearly independent. Then, process $C_j$:
    $T_i = A_i \cdot (R_1 R_2 ... R_{N/m}) = (a_{i1} ... a_{im}) \cdot \begin{pmatrix} c_1, c_{m+1}, ..., c_{N-m+1} \\ . \\ . \\ . \\ c_m, c_{2m}, ..., c_N \end{pmatrix} where 1 \leq i \leq n$
(3) Send $T_1, T_2 \ldots T_n$ to the receiver.
```

The Receiver Party B : IDA-Decode

INPUT : encoded vectors $T_1, T_2 \dots T_m$
OUTPUT : a block of data C_j

(1) Assume that the receiver receives T_1, $\mathrm{T}_2 \dots \mathrm{T}_m$

$T_1 = A_1 \cdot R_1, A_1 R_2 ..., A_1 \cdot R_{N/m}$

$T_2 = A_2 \cdot R_1, A_2 R_2 ..., A_2 \cdot R_{N/m}$

...

$T_m = A_m \cdot R_1, A_m R_2 ..., A_m \cdot R_{N/m}$

(2) Prepare for the calculation of R_1
Based upon $T_1 \dots T_m$, and formula (4.3), we can get:

$$A' \cdot \begin{pmatrix} c_1 \\ . \\ . \\ . \\ c_m \end{pmatrix} = \begin{pmatrix} A_1 \cdot R_1 \\ . \\ . \\ . \\ A_m \cdot R_1 \end{pmatrix} where\_A' = \begin{pmatrix} a_{11} ... a_{1m} \\ ... \\ ... \\ ... \\ a_{m1} ... a_{mm} \end{pmatrix} \tag{4.3}$$

(3) Since A' is invertible, we can calculate R_1:

$$R_1 = \begin{pmatrix} c_1 \\ . \\ . \\ . \\ c_m \end{pmatrix} = \begin{pmatrix} a_{11} ... a_{1m} \\ ... \\ ... \\ ... \\ a_{m1} ... a_{mm} \end{pmatrix}^{-1} \begin{pmatrix} A_1 \cdot R_1 \\ . \\ . \\ . \\ A_m \cdot R_1 \end{pmatrix} \tag{4.4}$$

(4) Repeating step 3, we can calculate $\mathrm{R}_2 \dots \mathrm{R}_{N/m}$.
(5) Reconstruct C_j:
$\mathrm{C}_j = \mathrm{R}_1 || \mathrm{R}_2 \dots || \mathrm{R}_{N/m}$ where || denotes concatenation.

Chapter 5

Performance Analyses, Experimental Results and Security Discussion

The efficiency and security of the proposed approaches are the first priority in this thesis and therefore they are evaluated carefully in this chapter one after the other. In the first section, performance is analyzed, discussed and compared. In the second section, experiments are implemented and results are demonstrated for group key schemes as TGDH-H, TGDH+ and periodical TGDH-ASAP, based upon the real-time group membership behaviour data set captured in the MBone. For the hybrid architecture and M2 algorithms, experiments are developed using Network Simulation 2 (ns-2). In both simulation experiments, the fault tolerance of contributory group key schemes is verified and a group key could be generated and managed in every group session. In the third section, the authentication methods utilized in the proposed approaches to protect the rekey messages are described and the security of these approaches is studied.

5.1 Performance Analyses

In this section, the performance-relevant criteria are analyzed for TGDH-H, TGDH+, and periodic TGDH-ASAP. The hybrid architecture, M2, and improved FEA-M have already been discussed in chapter 4 so they will not be studied again here. In the following, the metrics for computational cost, communication overhead and memory consumption will be identified first. Then, TGDH-H, TGDH+, TGDH-ASAP will be evaluated by these metrics.

5.1.1 Metrics for Performance Evaluation

Computational cost

Every group key scheme comprises a variety of cryptographic operations. To begin with, this thesis considers the performance evaluation for each operation. Then, the performance costs for each operation are accumulated to attain the total costs.

Previous experiments [39], [61] demonstrate that each cryptographic scheme needs to be processed within a certain period of time, which can be viewed roughly as the performance cost it demands compared with other schemes. Therefore, like other research [39], [53], [62] this thesis assumes that the performances of these cryptographic operations can be measured by timing. The experimental results referred to in this thesis are listed below.

An experiment result: for the SUN ultra 1/170 workstation, the processing timings for the hash, encryption/decryption, DH, digital signing and digital signing verification operations are 0.01ms, 0.01ms, 100ms, 200ms and 50ms respectively [61] if the key size is 1024 bits.

According to the results, the hash and encryption/decryption operations show an almost equivalent performance and both of them are about 0.001 times equivalent to a DH operation. Then, insight analyses demonstrate us that every DH key scheme comprises two exponential operations for every party. Therefore, the computational cost for the hash or encryption/decryption operation is 0.002 times that of an exponential operation. So, the number of exponential operations can be treated as the metric when comparing the computational cost of each group scheme which includes different cryptographic operations. The number of encryption/decryption and hash operations can be transferred into the number of exponential operations by a factor of 0.002. Notice that all Diffie-Hellman operations deployed in this thesis can be replaced with Elliptic Curve Diff-Hellman operations. Elliptic Curve Diff-Hellman operations are more efficient in terms of computation cost and more lightweight in terms of communication overhead than that of the regular DH. However, to compare with previous solutions such as TGDH, I used regular DH in this thesis.

Communication overhead:

The areas for evaluating communication overhead consist of the number of rounds, the number of unicasts and the number of multicasts. Previous research [15] shows that the impact of unicasts and multicasts on network bandwidth can be compared with respect to quantification. The costing function shown below was deployed by Chuang and Sirb [15].

$$R_{u/m} = \frac{L_u}{L_m} = n^{-0.8} \tag{5.1}$$

where n: group size; L_u*: average unicast hops,* L_m*: total hops of a multicast tree;*

This research uses it to evaluate the communication overhead between unicasts and multicasts. Utilizing formula (5.1), the number of unicasts can be transferred into the number of multicasts and finally each group key scheme is analyzed by comparing the number of multicasts it demands. Therefore, the number of the multicast is the metric for communication overhead for every group key scheme.

Memory consumption

In this thesis, for the sake of fairness, the key length for every group/auxiliary key should be the same. So, the metric for evaluating memory consumption is the number of group/auxiliary keys stored by every group member.

5.1.2 Performance Evaluation for Each Group Key Scheme

In this sub-section, this thesis first introduces the view of group membership events so that subsequent discussion is based upon the same event. Then, the notions of computational costs and communication overhead for the event are defined. Finally, the performances of TGDH-H and TGDH+ are discussed.

Group Session Model:

First, let us take a look at the procedures for a group session. Every group session can be treated as a sequence of group members joining and group members leaving.

Therefore, this thesis assumes that every group session is comprised of a set of *J Join & 1 Leave* ($J \geq 0$) events.

The performance for the *J Join & 1 Leave* ($J \geq 0$) scenarios, which are shown in Figure 5.1 (group member join/leave for TGDH+ and TGDH-H), is discussed. In Figure 5.1 both the key tree, T_{main}, and child key tree, T_{child}, are available. Assume that the number of members in T_{main} is n and the number of members in T_{child} is J. For the sake of simplification, assume that $n = 2^x$ and $J = 2^y$ where x and y are integers. Hence, both key trees are balanced.

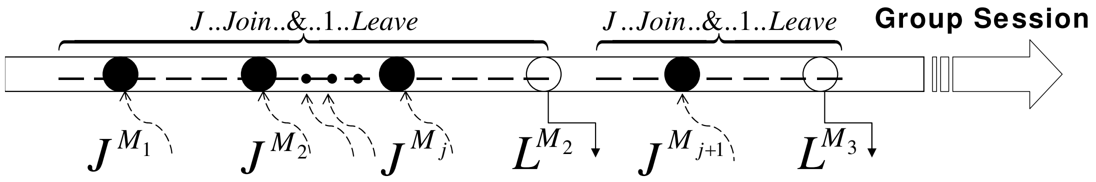

Figure 5.1: Group Session Model

<u>Computational Cost:</u>

Let *COMP(J, n)* denote the combined computational cost for all group members to update the group keys for one *J Join & 1 Leave* event. *COMP(J, n)* is comprised of the number of hash operations, the number of encryption/decryption operations, the number of DH operations and the number of digital signing operations.

$$COMP(J,n) = N_{J,n}^{SIGN} \quad and \quad N_{J,n}^{DH} \quad and \quad N_{J,n}^{ENC} \quad and \quad N_{J,n}^{Hash}$$

where $N_{J,n}^{Hash}$:number of Hash;$N_{J,n}^{ENC}$:number of encryption/decryption;

$N_{J,n}^{DH}$:number of Diffie-Hellman;$N_{J,n}^{SIGN}$:number of digital signing;

Communication Overhead:

Let *COMM(J, n)* denote the combined communication overhead for all group members to update the group keys for one *J Join & 1 Leave* event in which the original group size is n. *COMM(J, n)* is comprised of the number of unicasts and the number of multicasts.

$$COMM(J,n) = N_{J,n}^{Unicast} \quad and \quad N_{J,n}^{Multicast} where \begin{array}{l} N_{J,n}^{Unicast} : Numberof \quad Unicast; \\ N_{J,n}^{Multicast} : Numberof \quad Multicast \end{array}$$

TGDH-H:

The following is an analysis of the *J Join & 1 Leave* scenario, as shown in Figure 5.1.

Computational cost for TGDH-H:

For every single group member joining, each group member should use hash to update its group key. The total number of hash operations for updating all the group members' group keys for J join should be:

$$N_{J,n}^{Hash} = \sum_{i=1}^{J} (n + i - 1) = J(2n + J - 1)/2;$$

For every group member joining, in addition to the hash operations, encryption/decryption operations and ID-based Diffie-Hellman operations (for each party, three exponential operations are demanded) are also required while the sponsor delivers the group key to the new member. The total number of encryption/decryption operations and exponential operations for J joining should be:

$$N_{J,n}^{Encrypt} = \sum_{i=1}^{J} 2 = 2J; \quad N_{J,n}^{Expon.} = 2\sum_{i=1}^{J} 3 = 6J;$$

When one group member leaves, the leave protocol should update the keys on T_{child} and those on the key path for M$_k$. Keys associated with the leave operations on T_{child} are computed already in the case of the join protocol. The number of node keys to be updated by all the members in T_{child} is yJ and the number of blinded keys to be updated by all the members in T_{child} is $J-1$. To update the keys on M$_k$'s key path, the number of node keys to be updated by all the members should be $2n$ and the number of blinded keys to be updated by all the members should be x. In total, the number of exponential operations is:

$$N_{J,n}^{DH} = yJ + (J-1) + 2n + x = (y+1)J + 2n - 1$$

In the case where a group member joins, the join protocol does not demand any signing operations. In the case where a group member leaves, according to the *Dominating* algorithm, the number of multicast for updating T_{child} and the M$_k$ key path should be $J/2$.

$$N_{J,n}^{SIGN} = J/2$$

Consequently, it shows the potential that the device cannot respond in time if the rate for the group member join / leave is too high:

Estimated acceptable rate for join requests: rate $\leq 1/(2 * T_{DH}\ )$

Estimated acceptable rate for leave requests: rate $\leq 1/((3 * J + 2 * n - 2) * T_{DH})$

where T_DH is the timing of DH operation. Communication Overhead for TGDH-H: In the case where one group member joins, this join protocol uses an ID-based Diffie-Hellman authentication which sends two unicast messages to generate the shared key between the sponsor and the new group member. In the case where one group member leaves, according to the *Dominating* algorithm, the number of signing operations to update the T_{child} and the M_k key path should be $J/2$.

$$N_{J,n}^{Unicast} = 2J; \quad N_{J,n}^{Multicast} = J/2$$

TGDH+

The *J Join & 1 Leave* scenario, as shown in Figure 5.1 is analyzed below.

Computational cost for TGDH+:

For every group member joining, every member should use hash to update its group key and the sponsor should encrypt its hash result and send it to the new member. In the case where a group member joins, the join protocol demands DH, Hash and Encryption/Decryption operations. The join protocol for handling J joining requires 2J times the DH operations.

$$N_{J,n}^{Hash} = \sum_{i=1}^{J} (n + i - 1) = J(2n + J - 1)/2$$

$$N_{J,n}^{ENC} = \sum_{i=1}^{J} 2 = 2J; \qquad N_{J,n}^{DH} = \sum_{i=1}^{J} 2 = 2J$$

When one group member leaves, there are four cases, as discussed earlier.

Case 1: TGDH is used to handle this 0 join & 1 leave scenario.

$$N_{J,n}^{DH} = 2n - 1; \qquad N_{J,n}^{SIGN} = 1$$

Case 2: group members in T$_{main}$ should process hash operations. One DH is launched between a group member in T$_{main}$ and a group member in T$_{child}$. One encryption and one decryption is also needed between them. In T$_{child}$, keys associated with the leaves on T$_{child}$ are already computed in the case of the join protocol. All other DH-based keys should be updated and all group members should decrypt the new group key.

$$N_{J,n}^{SIGN} = J/2; \quad N_{J,n}^{DH} = 3J/2 + 1; \quad N_{J,n}^{ENC} = J + 2; \quad N_{J,n}^{Hash} = 2(n - J)$$

Case 3 or Case 4: the leave protocol should update the keys on T$_{child}$ and those on the key path for M$_k$. Keys associated with the leaves on T$_{child}$ are already computed in the case of the join protocol. So the number of keys to be updated by all members in T$_{child}$ is (J-1). The number of keys to be updated by all members in T$_{main}$ should be 2n-1 due to the updating of M$_k$'s key path.

$$N_{J,n}^{DH} = (J - 1) + 2n - 1 + 2J = 3J + 2n - 2$$

<u>Communication cost for TGDH+</u>:

In the case where one group member joins, this proposal's join protocol uses the ID-based Diffie-Hellman authentication which sends two unicast messages to generate the shared key between the sponsor and the new group member. In the case where one group member leaves, according to the *Dominating* algorithm, the number of signing operations to update T_{child} and the of M$_k$ key path should be $J/2$.

$$N_{J,n}^{Unicast} = 2J; \quad N_{J,n}^{Multicast} = J/2$$

When one group member leaves, there are 4 cases.

Case 1: $N_{J,n}^{Multicast} = 1$

Case 2: $N_{J,n}^{Unicast} = 2$; $N_{J,n}^{Multicast} = J/2$

Case 3 or Case 4: According to the dominating algorithm, the number of signing operations to update for updating T$_{child}$ and M$_k$'s key path should be J/2. This means that $N_{J,n}^{Multicast} = J/2$

TGDH-ASAP

Computational cost for TGDH-ASAP:

For every joining operation, each member should use hash to update its group key and the sponsor should encrypt its hash result and send it to the new member.

$$N_{J,L,n}^{Hash} = \sum_{i=1}^{J} (n + i - 1) = J(2n + J - 1)/2; \quad N_{J,L,n}^{ENC} = \sum_{i=1}^{J} 2 = 2J;$$

The join protocol for handling J join requests requires *2J* DH operations. At the end of the interval, the keys on T$_{child}$ and on other $L - 1$ key paths should be computed. Keys associated with the leaves on T$_{child}$ are already computed in case with the join protocol. So,

$$N^{DH}_{J,L,n} = L(\log_2 \tfrac{n}{L}) + 4J - L - 2$$

$$+2((J-L)\log_2(J-L) + n(2 + \log_2 L + 1))$$

When a group member joins, the join protocol requires no signing operations. At the end of the interval, according to the *ASAP* scheme, the number of signing operations is 1. So, $N^{SIGN}_{J,L,n} = 1$.

Communication Overhead for TGDH-ASAP:

For group member join requests, unicasts are required to achieve the group key scheme. At the end of the interval, unicasts and multicast are required as well for updating the key tree.

$$N^{Unicast}_{J,L,n} = 2J; \qquad N^{Multicast}_{J,L,n} = 1;$$

5.1.3 Performance Comparison for Group Rekey Schemes

Performance Comparison for Individual Rekeying

- TGDH vs. STR vs. TGDH-H vs. TGDH+

TGDH [25] and STR [26] have been known as the most efficient contributory group key management schemes to provide individual rekeying. Detailed comparison have been published [3], [25], [26]. The following is a comparison of TGDH-H and TGDH+ with TGDH and STR. In Table 5.1 and Table 5.2, the computational cost, communication overhead and memory consumption for TGDH-H, TGDH+, TGDH and STR are summarized. The current group size is denoted by n and the height of the key tree for TGDH, TGDH-H and TGDH+ is h. For merge protocols, the number of sub-groups is k and the number of group members in every sub-group is m. For partition protocols, the number of leaving members is p. In Table 5.1 and Table 5.2, both the total cost and the main sponsor's cost includes the cost for all the group members. For TGDH, TGDH-H and TGDH+, this performance analysis is based upon an average scenario. The overhead varies according to the balance of the key tree and the join or leave member's location in the key tree. In the following, the J join & 1 leave event, the merge protocol and the partition protocol are compared.

Table 5.1: Computational Cost

		Main sponsor			Total		
		Exponentiation	Hash/Encry	Signatures	Exponentiation	Hash/Encry	Signatures
TGDH	J join&1 leave	2h(J+1)	-	J+1	(2n-1)(J+1)	-	2J+1
	Merge	2h	-	Log_2k+1	2(h-$\log_2$k)k+(2k-1)	-	2k
	Partition	2h	-	Min($\log_2$p+1, h)	2(h-$\log_2$p)p+(2p-1)	-	min(2h,2p)
STR	J join&1 leave	4J+(3n/2+2)	-	J+1	(2n+2)J+(3n/2+2)	-	2J+1
	Merge	3m+1	-	2	(n+m)m+3m+1	-	k+1
	Partition	3n/2+2	-	1	(n-1) (3n/4+1) +3n/2+2	-	1
TGDH-H	J join&1 leave	2(h+$\log_2$J)	J+2	1	6J+4n-4	J(J+2n+1)	3J/2+1
	Merge	2h	-	1	2(h-$\log_2$k)k+(2k-1)	-	k
	Partition	2h	-	1	2(h-$\log_2$p)p+(2p-1)	-	p
TGDH+	J join&1 leave	2(h+$\log_2$J)	J+2	1	6J+4n-4	J(J+2n+1)	J/2+1
	Merge	2h	-	1	2(h-$\log_2$k)k+(2k-1)	-	k
	Partition	2h	-	1	2(h-$\log_2$p)p+(2p-1)	-	p

Table 5.2: Communication Overhead and Memory Consumption

		Rounds	Communication overhead				Memory
			Main sponsor		Total		
			Unicast	Multicast	Unicast	Multicast	
TGDH	J join&1 leave	2J+1	-	[1, 2J+1]	-	2J+1	0
	Merge	$\log_2$k+1	-	H	-	2k	0
	Partition	min($\log_2$p+1, h)	-	H	-	min(2h,2p)	0
STR	J join&1 leave	2J+1	-	2	-	2J+1	0
	Merge	2	-	1	-	k+1	0
	Partition	1	-	1	-	p	0
TGDH-H	J join&1 leave	2J+1	1	1	3J	J+1	0
	Merge	$\log_2$k+1	-	1	-	k	0
	Partition	min($\log_2$p+1, h)	-	1	-	1	0
TGDH+	J join&1 leave	2J+3	1	1	2J+2	J/2	[0, 1]
	Merge	$\log_2$k+1	-	1	-	k	0
	Partition	min($\log_2$p+1, h)	-	1	-	1	0

J join & 1 leave:

Computational costs

As seen from Table 5.1, TGDH+ is the most efficient in terms of the number of exponentiations and the number of signing operations. TGDH-H ranks second. STR and TGDH demand the most computational costs.

Communication costs

STR and TGDH require the most communication overhead; TGDH+ demands the least. TGDH+ requires two more rounds than TGDH and STR. However, what TGDH+ deploys for the two more rounds is a one-hop unicast. By contrast, other key management schemes, TGDH, STR, and TGDH-H, use multi-hop multicasts for every round which means more hop deliveries are required to send the rekey messages around the network.

Storage costs

TGDH, STR and TGDH-H require no more auxiliary group keys. Most members of TGDH+ should store not only a group key but an auxiliary group key.

Merge:

TGDH-H and TGDH+ have a lower cost compared with TGDH and STR in terms of the number of multicasts and computational costs. STR needs the most number of exponentiation operations and TGDH requires the most number of signing operations. However, STR uses the least number of rounds.

Partition:

TGDH demands the most communication overhead and the most signing operations.

STR requires the least number of rounds, the least number of signing operations and the least number of multicast messages, but it has the highest computational cost, O(n^2) times the exponentiations. Consequently, in terms of computational cost, TGDH-H and TGDH+ are more efficient.

Finally, for our purposes, TGDH-H and TGDH+, are more efficient in *J join & 1 Leave* and merge protocols. Specifically, TGDH+ requires the lowest computational costs, one more round and one more auxiliary group key compared with TGDH-H, TGDH and STR. For partition protocols, STR works better in signing and multicast metrics. For the remaining metrics of the partition protocol, TGDH-H and TGDH+ work better.

5.1.4 Experimental Results

MBone

Group communications were originally proposed to address the implementation of multicasts which lead to the development of a variety of collaborative applications. MBone (short for Multicast Backbone) is an example. As an early group communication service, MBone is an experimental backbone standing in for a virtual infrastructure providing multicast services for real-time data communications. In 1992, MBone was originated at the March meeting of the Internet Engineering Task Force (IETF) [11]. Equipped with Real-Time Protocol (RTP) [16] and Protocol Independent Multicasting (PIM) [52], MBone has seen significant progress since 1992. Consequently, MBone serves as a platform for multicast services and group conferencing as well.

In this sub-section, the real-time group membership behaviour captured in MBone is used to compare six individual group key management schemes, TGDH-ASAP, Queue batch, TGDH, STR, TGDH-H, and TGDH+. The timings need to generate/update the group key includes two parts, the processing time and the partial key transport time. As for the later, the longer the length, the more time should be taken to establish/renew the group key. However, considering that the time to transmit the signal around the planet is much shorter than that to perform heavyweight cryptographic operations for a regular device in resource-limited networks, the length of the communication channel affects the time for generating/updating the group key trivially. Consequently, it is not counted as an important factor in terms of rekey timings.

The *Tool for Collecting MBone Session Dynamics* project at the College of Computing, Georgia Institute of Technology developed the *Mlisten* data collection tool to capture information about when the group member joins and how long the duration of memberships lasts. For communication in MBone sessions, a multicast IP address is advertised for the purpose of addressing and two pairs of UDP ports are utilized for spread connections. One pair is the *data pair* used to deliver group communication data and the other pair is the *control pair* deployed to forward group member actions such as joining and leaving.

Almeroth and Ammar, used *Mlisten* to monitor the group member behaviours [1] [2]. Unreliability caused by data transport jitter or packet loss was corrected and abnormal MBone behaviour eliminated. As a result, accurate data collecting can be achieved in real-time.

Specifically, three data sets, (3.5) NASA STS-63 shuttle, (4.3) a UCB Multimedia Lecture Series audio session and (4.4) all audio sessions, have been recorded in the log file (ftp://ftp.cc.gatech.edu/people/kevin/release-dat) and posted online. The log data has been studied and it was found that the group size is around 100 for every session and the rate of join/leave requests is not high and is smooth [1] [2]. Although group member behaviour is gathered from MBone, researchers for group communications still believe that the group membership data can be treated as representative while understanding other group behaviour underlying such network settings as MANETs. To provide an example, a tiny portion of the original data is listed below.

multicast address Host address Date time Duration

+ port ID + port ID (join) (join) (s)

```
224.2.167.31:30758 129.13.81.2:1172 11/12/96 03:38:11 15
224.2.212.30:45145 129.13.81.2:1184 11/12/96 03:39:33 2
224.2.165.1:18278 129.13.81.2:1191 11/12/96 03:40:34 3
224.2.119.149/127:21732 129.13.81.2:1195 11/12/96 03:40:57 17
224.2.119.149:21732 129.13.81.2:1195 11/12/96 03:40:57 17
224.2.251.56:24654 129.13.81.2:1208 11/12/96 03:49:35 36
224.2.119.149/127:21732 129.13.81.2:1243 11/12/96 08:07:27 5
224.2.119.149/127:21732 129.13.81.2:21732 11/12/96 08:07:48 18
224.2.119.149:21732 129.13.81.2:21732 11/12/96 08:07:51 17
224.2.181.236:19562 129.13.81.2:19562 11/12/96 08:08:29 73
224.2.181.236:19562 129.13.81.2:1254 11/12/96 08:09:55 127
```

```
224.2.204.125:29862 129.13.81.2:29862 11/12/96 10:30:19 11302

224.2.251.56:24654 129.13.81.2:1286 11/12/96 10:33:22 69

224.2.0.1:23456 129.13.81.2:23456 11/12/96 10:34:57 36
```

In this proposal, in order to compare performance costs, two group sessions were selected randomly to explore and examine the key management schemes: TGDH, STR, TGDH-H and TGDH+. The first group session was with the advertised address “224.0.1.11” and the second “224.2.100.100”.

5.1.5 STR vs. TGDH vs. TGDH-H vs. TGDH+

In terms of computational cost, the number of exponentiations (hash and Encrypt/Decrypt operations is included by translating them into exponentiation by a ratio of 0.002) for different group sizes is listed in Figure 5.1 and 5.5 for session 1 and session 2 respectively. With regard to communication overhead, the total number of multicasts for every group session (unicasts are included by translating them into multicasts by a ratio of $n^{-0.8}$ where n is the group size) is listed in Figure 5.2 and 5.6 for session 1 and session 2 respectively. The total number of exponentiations for every group session is listed in Figure 5.3 and 5.7 for session 1 and session 2 respectively. The total number of signings for every group session is listed in Figure 5.4 and 5.8 for session 1 and session 2 respectively. The result shows that these proposals are more efficient than TGDH and STR with respect not only to computational cost but also to communication overhead. According to the result mentioned earlier, TGDH+ is computationally lighter than TGDH-H. After examing the data set of MBone, I realize that it is resulted from the scenario that a number of group members join the group and then leave it very soon due to lacking of interests for the group session. If every group member participates a group session in purpose and stays within the group session for long-term, TGDH-H will show better performance than TGDH+ since the auxiliary key method will not be utilized for this case. That will lower the performance of TGDH+.

Furthermore, after analyzing the key tree structure with the input of the real data of MBone, a skinny key tree has been encountered during the group key generation/management for TGDH. However, in contrast, key trees for TGDH-H and TGDH+ are more balanced than that of TGDH becasue of the utilization of moving child key tree.

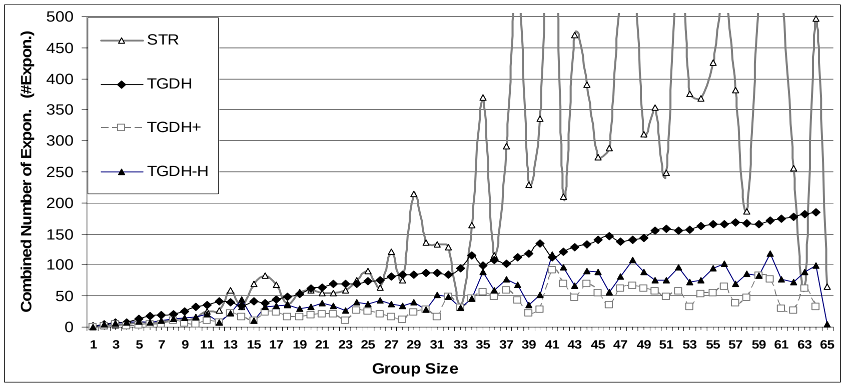

Figure 5.2: Individual Rekey: Number of Exponentiations for Session 1

5.1.6 Queue-Batch vs. TGDH vs. TGDH-ASAP

In terms of computational cost, the number of exponentiations (hash and Encrypt/Decrypt operations are included via translating them into exponentiation with a ratio of 0.002)

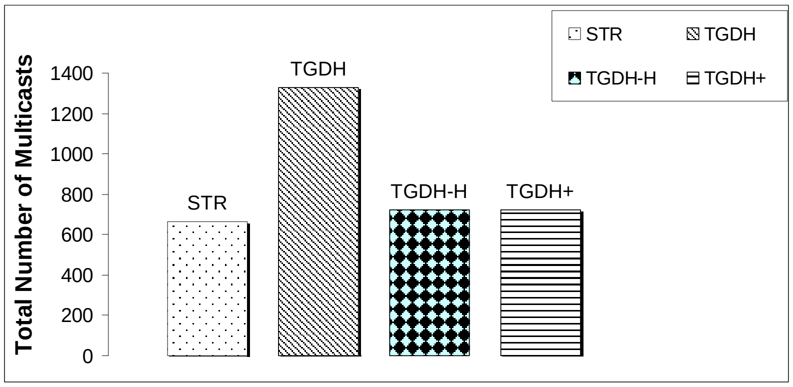

Figure 5.3: Individual Rekey: Total Number of Multicasts for Session 1

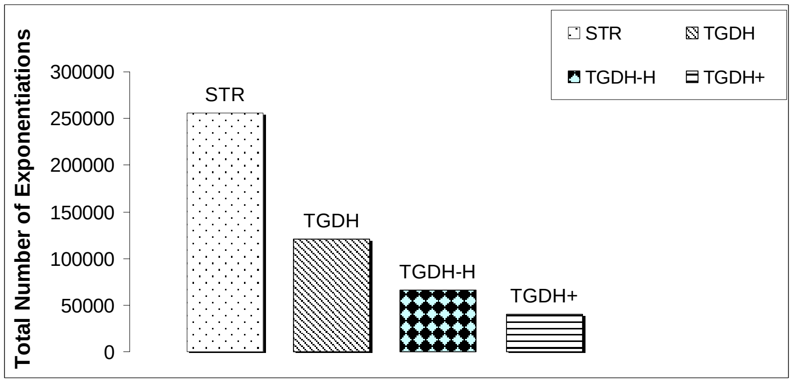

Figure 5.4: Individual Rekey: Total Number of Exponen. for Session 1

for different group sizes is listed in Fig. 5.9. The total number of exponentiations for this session is listed in Fig. 5.11. The total number of signings for every session is listed in Fig. 5.12. In terms of communication overhead, the total number of multicasts for every session (unicasts are included via translating them into multicast with a ratio of $n^{-0.8}$ where n is the group size) is listed in Fig. 5.10.

According to Fig. 5.9 and Fig. 5.11, this proposal requires slightly more exponentiations than Queue-Batch. However, considering the total computational cost, including exponentiation and signing, this proposal is more efficient. Fig. 5.10 confirms that the present proposal requires the least communication overhead. With regard

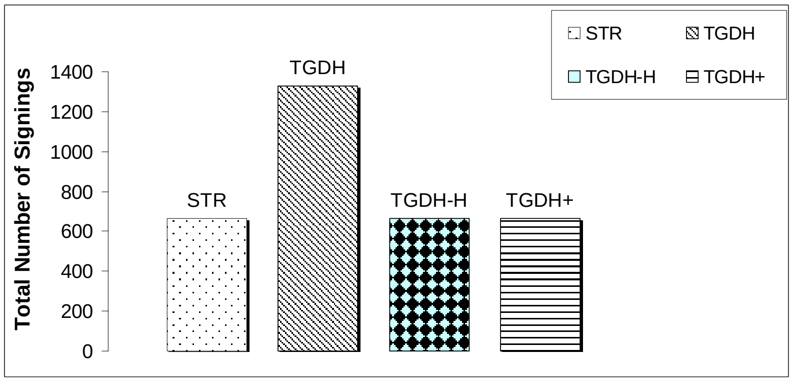

Figure 5.5: Individual Rekey: Total Number of Signing Operations for Session 1

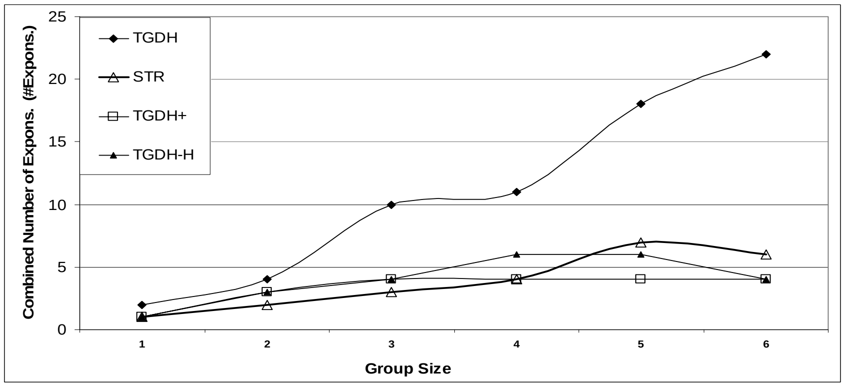

Figure 5.6: Individual Rekey: Number of Exponentiations for Session 2

to quality of service, this proposal responds to join requests immediately, whereas Queue-Batch does not admit new members until the end of the interval.

5.1.7 Hybrid Architecture

In this sub-section, hybrid architecture and straightforward methods are simulated via Network Simulation-2 (ns-2) [79], a widely used simulation tool. In this test, to achieve a routing function, AODV, a routing protocol, is deployed to connect wireless nodes and forward packets from one node to another. A multicast AODV (MAODV)

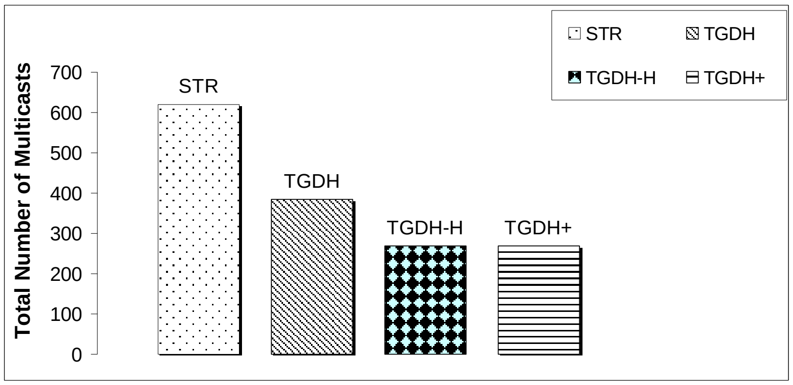

Figure 5.7: Individual Rekey: Total Number of Multicasts for Session 2

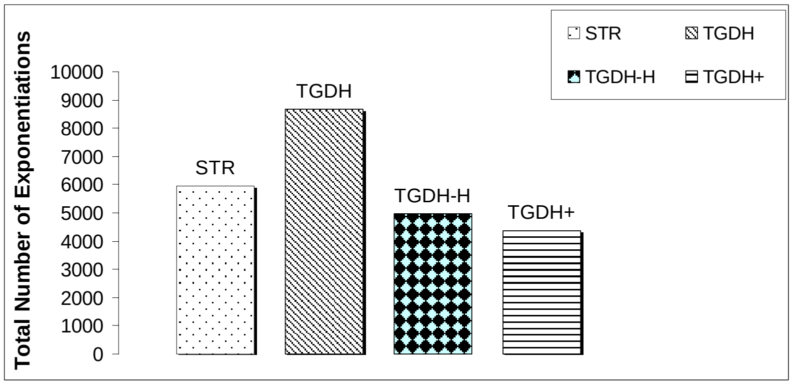

Figure 5.8: Individual Rekey: Total Number of Exponen. for Session 2

module [80] is extended at ns-2 to multicast packets for this project. Specifically, this simulation utilized the test scenario components listed below:

NS2 version: ns-2.27

Network: Mobile Ad Hoc Network (MANET)

Routing Protocol: AODV

Multicast Protocol: MAODV

Area: 1500 x 300 meters

Number of nodes: 50

Number of repetitions: 10

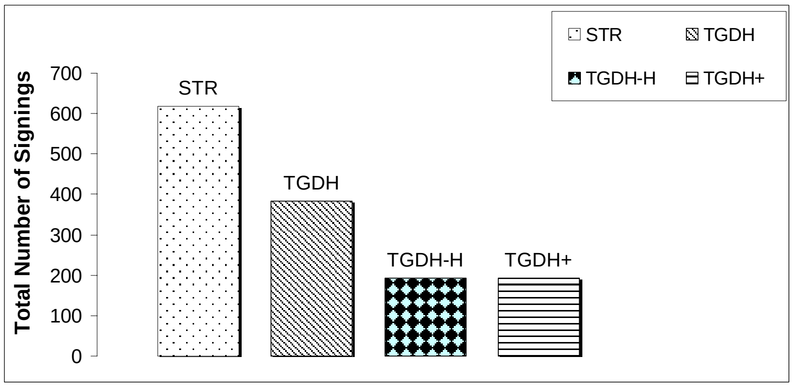

Figure 5.9: Individual Rekey: Total Number of Signing Operations for Session 2

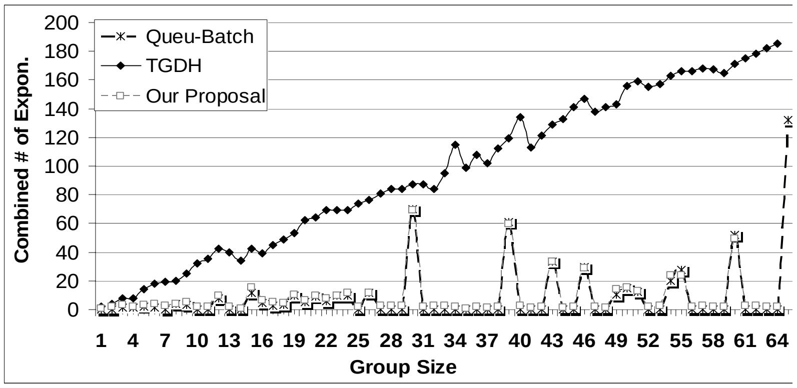

Figure 5.10: Periodical Rekey: Number of Exponentiations

Physical/Mac layer: IEEE 802.11 at 2 Mbps, 250 meter transmission range

Mobility model: random waypoint model with no pause time, maximum speed 20 m/s(high mobility scenarios).

This test was developed to simulate a scenario in which the key server is out-of-service, in order to contrast how the straightforward method and the proposed hybrid architecture generate a new group key. Specifically, the hybrid architecture multicast the message to notify other group members that the key sever is out of service if the key server cannot be detected with heartbeats. Then, a notification to let all group members switch from the centralized method to the contributory one is forwarded as

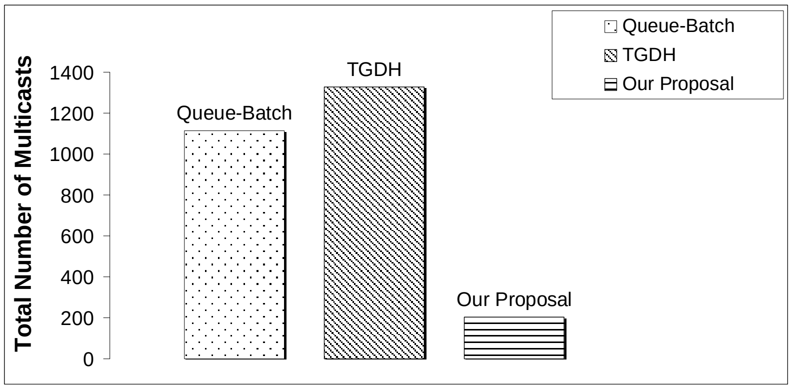

Figure 5.11: Periodical Rekey: Total Number of Multicasts

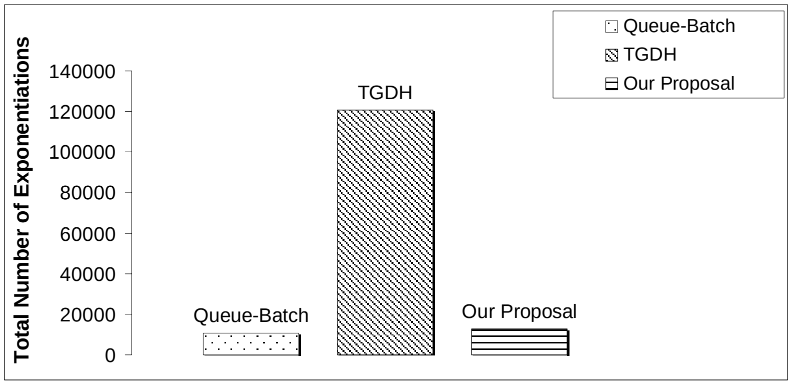

Figure 5.12: Periodical Rekey: Total Number of Exponentiations

well to other group members. By contrast, the straightforward method will generate a new key via contributory solutions.

According to the test result, the hybrid method is better than the straightforward method in terms of communication overhead. In terms of computational cost, the hybrid architecture requires no computational operations and the straightforward solution needs many operations to generate a new contributory group key.

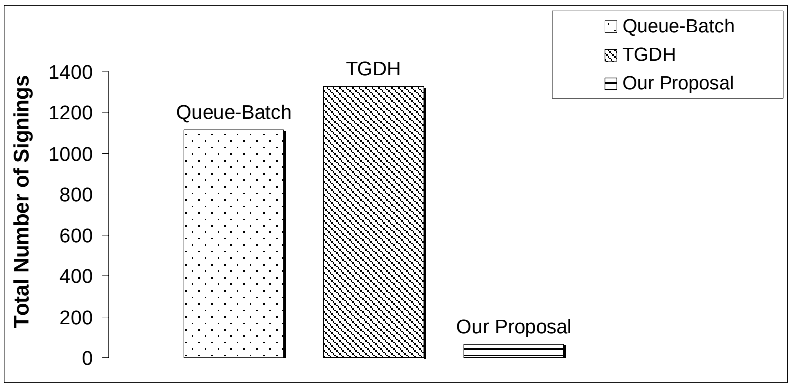

Figure 5.13: Periodical Rekey: Total Number of Signing Operations

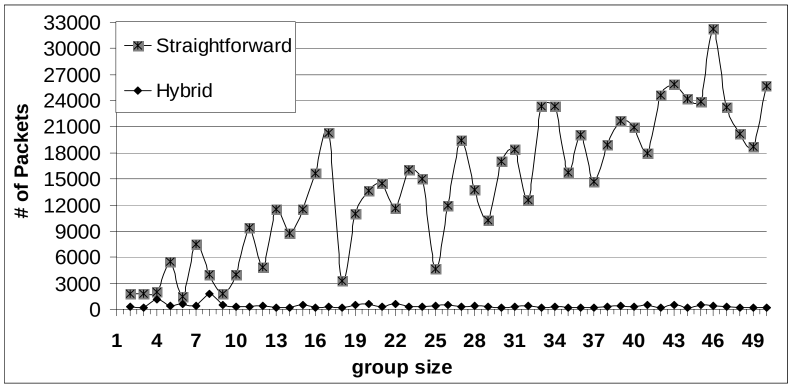

Figure 5.14: Communications Overhead: Hybrid vs. Straightforward

5.1.8 DST-M2 vs. SP-GDH

The simulation platform and configations are the same as in section 5.1.6. The result is demonstrated in Figure 5.15:

5.2 Security Discussion

This sub-section introduces authentication techniques to protect the rekey messages. Then, security issues for TGDH-H, TGDH+ and TGDH-ASAP are discussed.

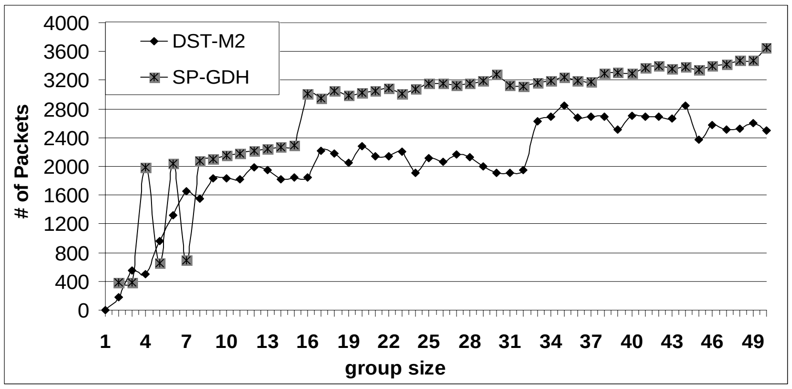

Figure 5.15: Communication Overhead: DST-M2 vs. SP-GDH

5.2.1 Authentication Protocols

In this thesis, rekey messages are delivered around the network via unicast or multicast technology. To resist man-in-the-middle attacks, three authentication techniques are provided.

Unicast.

The join and leave protocol utilizes unicast technology. When one group member joins the group, the sponsor should generate a secure channel between them and deliver the new member the updated group key. To set up the secret channel, an ID-based Diffie-Hellman key exchange scheme [71] is used to generate the shared key between the sponsor and the new member. Then, AES [32] is utilized to encrypt the new group key with the shared key. Finally, the ciphertext is sent from the sponsor to the new member via unicast delivery, to authenticate which, Hash-based Message Authentication Codes (MAC) known as HMACs [32] are deployed. Having already obtained the shared key, the new group member can decrypt the ciphertext and check its integrity via HMACs. It is similiar for the case 2 leave protocols for TGDH-H and

TGDH+.

Multicast:

The leave, merge and partition protocols utilize multicasts to send rekeying messages. To authenticate and encode multicast messages, the Signature Amortization Information Dispersal Algorithm (SAIDA) [41] is used which signs the hash output of the entire multicast message with RSA. RSA is well known for its efficient verification operations. Therefore, SAIDA [41] can reduce the number of signing operations and provide a low computational cost for verification. Other group members, having received the message, can verify the data's original authentication and update their group key.

Mutual authentication:

During group membership changes, mutual authentication is necessary to guarantee two parties of the scheme are exactly who they claim to be. Current available solutions for specific networks can be deployed which means that this thesis assumes the previous installment of default authentication schemes. For example, in Windows Mobile Operating Systems or Windows Vista operating systems, a variety of authentication schemes such as certificates and computer/user Kerbors V5, are utilized for authentication purposes. With the installation of such Operating Systems, one join/leave member and one insider member can confirm each other's legal status as long as the former needs to join or asks to leave. By using this solution, there will be no problems for MIM or illegal actions.Where authentication services, such as Kerb V5, are not deployed on-line, an ID-based solution can be used instead to provide authentication services.

5.2.2 Security Analyses

The security of these individual group rekey schemes is based upon the security assumptions of the two-party Decision Diffie-Hellman problem (DDH) [32], one-way hash function (Hash) [32] and the Decision Binary Tree Diffie-Hellman problem (DBTDH) [26], the details of which are well established [26] [32]. In this thesis, forward secrecy, backward secrecy and loosely equivalently key independence for all the proposed group key schemes (TGDH-H, TGDH+ and TGDH-ASAP) can be guaranteed and the security argument for them is provided below.

Firstly, let us consider backward secrecy. When a new group member joins, the group key is updated via the one-way hash function for group members in TGDH-H, TGDH+ or TGDH-ASAP. It is not computationally feasible to deduce the input of the hash function even when the output of the hash function is known. So, the new group member cannot determine the previous group keys. Furthermore, the new group member will be forwarded the current group key encrypted by an intermediate secret shared between the new member and the sponsor. Therefore, the current group key cannot be known by any outsiders. Consequently, backward secrecy can be guaranteed.

Secondly, forward secrecy guarantees that the leaving member cannot compromise any future group keys. For case 1, 3 and 4 of the leave protocol for TGDH-H and TGDH+, the node keys and blinded keys on these changed key paths are refreshed or updated. It is clear that this method belongs to the DBTDH problem, the backward secrecy property of which has already been proven [25]. For case 2, the leave protocol for TGDH+ uses the one-way hash function scheme. As discussed earlier, the auxiliary group key which is used to calculate the new group key is not known by the leaving member. Consequently, the leaving member cannot compromise the new group key. After each group member leaves, the auxiliary group key is changed so that no member belonging to the child key tree can obtain the new auxiliary group key. The same applies to the TGDH-H leave protocol.

Thirdly, key independence can be satisfied with this protocol as well since it is not computationally feasible to compromise the hash function scheme. Also, key material is delivered within a secure channel (SAIDA to protect multicast; ID-based Diffie-Hellman and HMAC to protect unicast). An outsider cannot compromise this key material since the RSA signing and HMAC techniques are too difficult to be broken. The one exception occurs when the group key is known by an outsider who knows the following group keys via hash function until a group member leaves. Although this loose key independence is kind of violent the independence discipline, this tradeoff is worth considering for the performance gains.

Fourthly, authentication needs to be discussed. As analyzed earlier, utilizing authentication solutions to protect the unicasts and multicasts can guarantee that the proposed group key scheme is not vulnerable to man-in-the-middle and impression attacks. Denial of Service (DoS) attacks are well studied and are beyond the scope of this thesis. These proposals can utilize well-known solutions [81] which protect the process against DoS.

Chapter 6

Conclusion

6.1 Summary of Research Work

Secure group key schemes play a key role in such group applications as collaborative tasks, network games and multimedia conferences. In this research, three efficient group key management schemes, an efficient group key partial key delivery technique, a hybrid group key management architecture, and improved encryption algorithms to facilitate the deployment for group key & encryption mechanisms were addressed. Specifically, proposed approaches included two efficient individual rekey schemes (TGDH-H and TGDH+) which can process join requests with a one-way hash function and postpone exponential operations when group members leave. Other methods including moving key trees and utilizing auxiliary keys reduce computational cost and communication overhead significantly. Furthermore, an efficient periodic batch rekey scheme (TGDH-ASAP) has been proposed to overcome the out-of-sync problem resulting from the proposed individual rekey schemes when the rate of group member join/leave requests is too high to be handled. In addition, maximum matching algorithms (M2) were proposed for decreasing communication overhead in generating contributory group keys. Furthermore, a hybrid group key management architecture has also been designed which combines the advantage of the centralized approach's efficiency and the contributory scheme's fault-tolerance. The hybrid architecture incorporates both a centralized and a contributory scheme for key management, introducing a low cost switch. Finally, further vulnerabilities for FEA-M with appropriate solutions have been presented.

Performance analysis shows that the proposed techniques decrease the number of messages, the number of exponentiations or the number of signing operations compared with other popular key generation and management schemes.

Future research might include group key schemes against Byzantine attacks in case where the insider group members are not trustworthy.

6.2 Integrated Solution

Image a scenario with a number of nodes in MANETs. Several of the nodes are launching a group session to accommodate a peer-to-peer tele-conference. At the beginning of the session, lots of members are treated as initial members who cooperate with each other to create the group at a certain time. Then, nodes can join or group members can leave at any time as long as the group session is not closed.

Therefore, at the beginning of the group session, M2 algorithms can be used to deliver the partial keys to construct the key tree and the TGDH+/TGDH-H group key scheme is deployed to generate the group key. TGDH+/TGDH-H manages the group key until an out-of sync problem is triggered. If the rate of join/leave request is too high, TGDH-ASAP can be utilized to manage the group key updates. Furthermore, if the key server is online, the hybrid architecture can be used and the centralized scheme can switch to contributory ones in cases for which the key server is out of control / not available. For secret messages delivered among group members, the improved FEA-M algorithms can be used to encrypt/decrypt the messages.

Bibliography

[1] K. C. Almeroth and M. H. Ammar, Multicast Group Behavior in the Internet's multicast backbone (MBone), IEEE Communication Magazine, Vol 35 (6), pp. 124-129, June 1997.

[2] K. C. Almeroth, A long-term analysis of growth and usage patterns in the multicast backbone (MBone). In Proceedings of IEEE Conf. on Computer Communications (INFOCOM00), vol. 2, pp. 824-833, Tel Aviv, Israel, March 2000.

[3] Y. Amir, Y. Kim, C. Nita-Rotaru, J. Schultz, J. Stanton, and G. Tsudik, Secure Group Communication Using Robust Contributory Key Agreement. IEEE Trans. On Parallel and Distributed Systems. Vol. 15 (5), pp. 468-480, May 2004.

[4] E. R. Anton, and O. C. Duart, Group key establishment in wireless ad hoc networks. In Proceedings of Workshop on Quality of Service and Mobility (WQoSM 2002), pp. 11-17, Angra dos Reis, RJ, Brazil, Nov., 2002.

[5] G. Ateniese, M. Steiner, and G. Tsudik, Authenticated Group Key Agreement and Friends, In Proceedings of 5TH ACM Conf. on Computer and Communication Security (CCS98), pp. 17-26, San Francisco, California, United States, 1998.

[6] G. Ateniese, M. Steiner, G. Tsudik, New multiparty authentication services and key agreement protocols. IEEE Journal of Selective Areas Communication. Vol. 18, no. 4, pp. 628-639, April, 2000.

[7] C. Becker and U. Wille, Communication complexity of group key distribution, In Proceedings of the 5th ACM Conference on Computer and Communications Security (CCS 98), pp. 1-6, San Francisco, California, United States, Nov. 1998.

[8] D. Boneh, G. Durfee, and M. Franklin, Lower bound for multicast message authentication. In proceedings of the conference on advances in cryptography (EUROCRYPT01), pp. 437 452, Aarhus, Denmark, May 2001.

[9] E. Bresson, O. Chevassut, D. Pointcheval, and J.-J. Quisquater, Provably authenticated group Diffie-Hellman key exchange. In Proceedings of 8th

ACM Conf. on Computer and Communication Security (CCS01), pp. 255-264, Philadelphia, PA, USA, 2001.

[10] M. Burmester and Y. Desmedt, A secure and efficient conference key distribution system. In Proceedings of advances in Cryptology (EUROCPYPT 94), LNCS Vol. 950, pp. 275-286, 1994.

[11] S. Casner and S. Deering, First IETF Inetrnet audiocast, ACM Computer Communication Review, Vol 27 (7), pp. 92-97, July 1992.

[12] Y. Challal and H. Seba, Group key management protocols: a novel taxonomy, International journal of information technology, Vol. 2 (1), pp. 105-118, 2005.

[13] R. Chandra, V. Ramasubramanian, and K. Birman, Anonymous gossip: improving multicast reliability in mobile ad-hoc networks. In Proceeding of 21th International Conference on Distributed Computing Systems (ICDCS01), pp. 275283, Phoenix, Arizona, USA, April 2001.

[14] T. Chiang and Y. Huang, Group keys and the multicast security in ad hoc networks. In Proceedings of International Conference on Parallel Processing Workshops (ICPP 03), 385 390, Kaohsiung, Taiwan, Oct. 2003.

[15] J. Chuang and M. Sirbu, Pricing multicast communication: A cost based approach, in Telecommunication System, Vol 17 (3), pp. 281-293, 2001.

[16] S. Deering, D. Estrin, D. Farinacci, V. Jacobson, G. Liu and L. Wei, PIM architecture for wide-area multicast routing, IEEE/ACM Transaction on Networking, Vol 4 (2), pp. 153-162, Apr. 1996.

[17] J. Edmonds, Paths, trees and flowers, Canadian Journal of Mathematics. Vol 17, pp. 449-467, 1965.

[18] A. Fekete, N. Lynch, and A. Shvartsman, Specifying and using a partionable group communication service, IEEE/ACM Transaction on Networking, Vol 19 (2), pp. 171-216, May 2001.

[19] W. Goddard, S. T. Hedetniemi, D. P. Jacobs, and P. K. Srimani. Self-stabilizing protocols for maximal matching and maximal independent sets for ad hoc networks. In Proceedings of the 17th International Parallel and Distributed Processing Symposium, (IPDPS03), pp. 87-96, Nice, France, April, 2003.

[20] T. Gopalsamy, M. Singhal, D. Panda, and P. Sadayappan, A reliable multicast algorithm for mobile ad hoc networks. In Proceedings of the 22th International Conference on Distributed Computing Systems (ICDCS 02), pp. 563-570, Vienna, Austria, July 02-05, 2002.

[21] J. Gross, and J. Yellen. Graph theory and its applications. CRC Press, 1999.

[22] I. Ingemarsson, D. T. Tang, and C. K. Wong, A conference key distribution system, IEEE Trans. Information Theory, Vol. 28, no.5, pp. 714-720, Sept. 1982.

[23] P. Judge and M. Ammar, Security Issues and solutions in Multicast Content Distribution: A Survey, IEEE Network, Vol 17 (1), pp. 30-37, 2003.

[24] K. P. Kihlstrom, L. E. Moser, and P. M. Melliar-Smith, The SecureRing Protocol for Securing Group Communications. In Proceedings of the IEEE 31th Hawaii International Conference on System Science, (HICSS98), Vol. 3. pp. 317-326, Hawaii, USA, Janauary 1998.

[25] Y. Kim. A. Perrig and G. Tsudik. Simple and Fault-Tolerant Key Agreement for Dynamic Collaborative Groups. In Proceedings of the 7th ACM Conference on Computer and Communications Security (CCS00), pp. 235-24, Athens, Greece, Nov., 2000.

[26] Y. Kim. A. Perrig and G. Tsudik. Group Key Agreement Efficient in Communication, IEEE Transaction on Computers, Vol. 53, (7), pp. 905-921, July 2004.

[27] Patrick P. C. Lee, John C. S. Lui, and David K.Y. Yau. Distributed Collaborative Key Agreement and Authentication Protocols for Dynamic Peer Groups. IEEE/ACM Transactions on Networking, Vol. 14(2), pp. 263-276, April, 2006.

[28] Shujun Li and Kwok-Tung Lo, Security Problems with Improper Implementations of Improved FEA-M, accepted by Journal of system and software in May 2006.

[29] J. Luo, P. T. Eugster, and J.-P. Hubaux, Route driven gossip: Probabilistic reliable multicast in ad hoc networks, in Proc. of IEEE Conf. on Computer Communications (INFOCOM'03), pp.2229-2239, San Francisco, CA, March 2003.

[30] Mark S. Manasse, A survey of micropayment technologies, and the MilliCent system http://www-db.stanford.edu/infoseminar.Archive/SpringY99/spring99/manasse-slides.

[31] P. McDaniel, A. Prakash and P. Honeyman, Antigone: A flexible framework for secure group communication. In Proceedings of 8th USENIX Security Symposium, pp. 99-114, Washington, D.C., USA, August 1999.

[32] A. J. Menezes, P. C. V. Oorschot, and S. A. Vanstone, Handbook of Applied Cryptography, CRC Press, Oct. 1996.

[33] M. J. Mihaljevic and R. Kohno, Cryptanalysis of fast encryption algorithm for multimedia FEA-M, IEEE Communications Letters, Vol. 6(9), pp. 382 84, Sep. 2002.

[34] M. J. Mihaljevic and R. Kohno, On wireless communications privacy and security evaluation of encryption techniques. In Proceedings of IEEE Wireless Communication and Networking Conf. (WCNC02), Orlando, FL, USA, pp.865-868. Mar. 2002.

[35] M. J. Mihaljevic, On Vulnerabilities and Improvements of Fast Encryption Algorithm for Multimedia FEA-M, IEEE Trans. On Consumer Electronics, Vol. 49, pp, 1199-1207, Nov. 2003.

[36] S. Miner and J. Staddon, Graph-based authentication of digital streams. In Proc. of IEEE Symposium on Research in Security and Privacy, pp. 232-246, Oakland, CA, US, May 2001.

[37] L. Moser, Y. Amir, P. Melliar-Smith, and D. Agarwal, Extended Virtual Synchrony. In Proc. of IEEE Intl Conf. Distributed Computing Systems (ICDCS94), pp. 56-65, Poznan, Poland, June 1994.

[38] M. J. Moyer, J. R. Rao and P. R. Ohatgi, A survey of security issues in multicast communications, IEEE Network, Vol 13 (6), pp. 12-23, Dec, 1999.

[39] M. Okabe, S. Sakane and K. Miyazawa, A study of security architecture for control networks over IP, 1st international workshop on Networked Sensing Systems (INSS04), pp 128-133, Tokyo, Japan, 2004.

[40] B. Ouyang, X. Hong and Y. Yi, A Comparison of Reliable Multicast Protocols for Mobile Ad Hoc Networks, IEEE SoutheastCon 05, pp. 339-344, Fort Lauderdale, FL, US, April, 2005.

[41] J. M. Park, E. K. P. Park, and H. J. Siegel, Efficient Multicast Stream Authentication Using Erasure Codes, ACM Trans. On Information and System Security, Vol 6, no.2, pp. 258-285, May 2003.

[42] A. Perrig, D. Song, and J. D. Tygar. ELK, A new protocol for efficient large-group key distribution. In Proceedings of the IEEE Symposium on Security and Privacy, pp: 247-262, Oakland, CA, US, 2001.

[43] M. Rabin, Efficient dispersal of information for security, load balancing, and fault tolerance, J. ACM, Vol. 36 (2), pp. 335-348, 1989.

[44] S. Rafaeli and D. Hutchison, A survey of key management for secure group communication, ACM Computing Survey, Vol. 35 (3), pp. 309-329, Sep. 2003.

[45] V. Rajendran, Y. Yi, K. Obraczka, S. J. Lee, K. Tang and M. Gerla, Reliable, Adaptive, Congestion-Controlled Adhoc Multicast Transport Protocol: Combining Source-based and Local Recovery. Univerity of California Los Angeles, Technical Report, 2003.

[46] M. K. Reiter, Secure agreement protocols: reliable and atomic group mutlicast in RAMPART, in Proceedings of the 2nd ACM Conference on Computer And Communication Security (ACM CCS 94), pp. 68-80, Fairfax, Virginia, USA, 1994.

[47] K.-H. Rhee, Y.-H. Park and G. Tsudk, A group key management architecture for mobile ad-hoc networks, Journal of Information science and engineering, Vol. 21, pp. 415-428, 2005.

[48] L. Rizzo and L. Vicisano, RMDP: an FEC-based Reliable Multicast Protocol for Wireless Environments, ACM Mobile Computing and Communications Review, Vol 2(2), pp. 23-31 Apr. 1998.

[49] O. Rodeh, K. Birman, M. Hayden, Z. Xiao, and D. Dolev, Ensemble Security, Tech. Rep. TR98-1703, Cornell University, Department of Computer Science, September 1998.

[50] O. Rodeh, K. Birman, M. Hayden, and D. Dolev, Using AVL trees for fault tolerant group key management, Tech. Rep. 2000-1823, Cornell University, Department of Computer Science, 2000.

[51] O. Rodeh, K. Birman, M. Hayden, and D. Dolev, The architecture and performance of security protocols in the Ensemble Group Communication System, ACM Transactions on Information and System Security, Vol. 4 (3), pp. 289-319, August 2001.

[52] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson, RTP, A transport protocol for real-time applications, Tech. Rep. RFC 1889, IETF, Jan. 1996.

[53] S. Setia, S. Koussih, and S. Jajodia, Kronos: A scalable group re-keying approach for secure multicast. In Proceedings of the IEEE Symposium on Security and Privacy, pp. 215 228, Oakland, CA, US, May, 2000.

[54] A. T. Sherman and D. A. Mcgrew, Key establishment in large dynamic groups using one-way function trees. In Software Engineering, IEEE Trans. vol. 29, (5), pp. 444-458, May 2003.

[55] M. Steiner, G. Tsudik, and M. Waidner, Key agreement in dynamic peer groups, IEEE Transactions on Parallel and Distributed Systems, vol. 11(8), pp. 769-780, August, 2000.

[56] D. Steer, L. Strawczynski, W. Diffie, and M. Wiener, A secure audio teleconference system, Advances in Crytology (CRYPTO 88), pp. 520-528, Santa Barbara, California, USA, Aug. 1988.

[57] Y. Sun and K. J. Ray Liu, Securing Dynamic Membership Informationin Multicast Communications, in Proc. of IEEE Conf. on Computer Communications,

(INFOCOM04), Hongkong, China, Mar., 2004.

[58] K. Tang, K. Obraczka, S. J. Lee, M. Gerla,A reliable, congestion-controlled multicast transport protocol in multimedia multi-hop networks. In Proceedings of the 5th International Symposium on Wireless Personal Multimedia Communications, (WPMC 02), pp. 252-256, Honolulu, USA, October 2002.

[59] H. S. Venter and J. H. P. Eloff, A taxonomy for information security technologies, Computers and Security, vol. 22, (4), pp. 299-307, May 2003.

[60] R. Vitenberg, I. Keidar, G. V. Chockler, and D. Dolev, Group communication specifications: a comprehensive study, ACM Computing Survey, Vol. 33(4), pp. 427-469, Dec. 2001.

[61] M. Waldvogel, G. Garonni, D. Sun, D. Weiler and B. Plattner. The VersaKey Framework: Versatile Group Key Management, IEEE Journal of Selected Area on Communication. (Special Issue on Middleware) Vol. 17(9), pp. 1614-1631, Sep. 1999.

[62] X. Li, Y. Wang, O. Frieder, Efficient Hybrid Key Agreement Protocol for Wireless Ad Hoc Networks. In Proceedings of International Conference Computer Communications and Networks, (ICCCN02), pp. 404 409, Maimi, FL, USA, 2002.

[63] H. Weatherspoon, C. Wells, P. Eaton, B. Zhao, and J. Kubiatowicz. Silverback: A Global-Scale Archival System, Technical Report UCB/CSD-01-1139, Computer Science Division, University of California, Berkeley, CA., 2001.

[64] C. K. Wong and S. S. Lam. Digital signatures for flows and multicasts, IEEE/ACM Transactions on Networking, Vol 7 (4). pp. 502-513, Aug. 1999.

[65] C. K. Wong, G. Gouda, and Lam S. S., Secure group communications using key graphs, IEEE/ACM Trans. Networking, vol. 8, (1),pp. 16-30, Feb 2000.

[66] M. Yajnik, S. Moon, and D. Towsley. Measurement and modeling of the temporal dependence in package loss. In Proc. of IEEE Conf. on Computer Communications (INFOCOM 99), pp. 345-352, New York, USA, 1999.

[67] Wen-Her Yang and Shiuh-Pyng Shieh, Secure key agreement for group communications, International Journal of Network management, vol 11 (6), pp. 365-374, 2001.

[68] Y. R. Yang, X. S. Li, X. B. Zhang, and S. S. Lam. Reliable Group Rekeying: A Performance Analysis. In proceedings of ACM SIGCOMM01, pp: 27-38, San Diego, CA, USA, August, 2001.

[69] H. Yang, H. Luo, F. Ye, S. Lu, L. Zhang, Security in mobile ad hoc networks challenges and solutions IEEE Wireless Communications, Vol. 11 (1), pp. 38-47, Feb. 2004.

[70] X. Yi, C. H. Tan, C. K. Siew, and M. R. Syed, Fast encryption for multimedia, IEEE Trans. On Consumer Electronics, vol. 47, pp. 101-107, Feb. 2001.

[71] X. Yi, C. H. Tan, C. K. Siew, and M. R. Syed, ID-based key agreement for multimedia encryption, IEEE Trans. On Consumer Electronics, vol. 48, pp. 298-303, May, 2002.

[72] X. B. Zhang, S. S. Lam, D-Y Lee, and Y. R. Yang. Protocol Design for scalable and reliable Group Rekeying, IEEE/ACM Transactions on Networking, vol. 11, (6), pp. 908-922, Dec. 2003.

[73] Y. Zheng, Shortened digital signatures, signcryption and compact and unforgeable key agreement schemes, Submission to IEEE P1363a: Standard Specifications for Public-Key Cryptography, 1998.

[74] X. K. Zou, B. Ramamurthy, S. S. Magliveras, Secure Group Communication over data Networks, Springer, 2005.

[75] Multiprecision Integer and Rational Arithmetic C/C++ Library, http://indigo.ie/ mscott/, Shamus Software Ltd.

[76] Jazzpazzle. Multiprecision Integer and Rational Arithmetic C/C++ Library, http://indigo.ie/ mscott/, Shamus Software Ltd.

[77] Random Walk. Multiprecision Integer and Rational Arithmetic C/C++ Library, http://indigo.ie/ mscott/, Shamus Software Ltd.

[78] Liu's routing protocol. Multiprecision Integer and Rational Arithmetic C/C++ Library, http://indigo.ie/ mscott/, Shamus Software Ltd.

[79] http://www.isi.edu/nsnam/ns

[80] E. M. Royer and E. M. Perkins, Multicast Ad hoc On-Demand Distance Vector (MAODV) Routing, IETF, Internet Draft: draft-ietf-manet-maodv-00.txt, 2000.

[81] C. R. Lin, QoS Routing in ad hoc wireless networks, pp. 31-40, Local Computer Network, 1998, LCN '98 proceedings, 23rd Annual Conference.

[82] M. Ruthmair and G. R. Raidl. A Kruskal-Based Heuristic for the Rooted Delay-Constrained Minimum Spanning Tree Problem, Vol. 5717, pp. 713-720, Lecture Notes in Computer Science, 2009.

[83] H. Bast, K. Mehlhorn, G. Schafer and H. Tamaki. Matching Algorithms Are Fast in Sparse Random Graphs, Vol. 39, No. 1, pp. 3-14, Theory of Computing Systems, Feburary, 2006.