NEW DYNAMIC GROUP KEY AGREEMENT PROTOCOLS AND THEIR FORMAL ANALYSIS AND APPLICATIONS

by

Orhan Ermiş

B.S., Computer Engineering, Bahçeşehir University, 2005

M.S., Computer Engineering, Bahçeşehir University, 2007

Submitted to the Institute for Graduate Studies in
Science and Engineering in partial fulfillment of
the requirements for the degree of
Doctor of Philosophy

Graduate Program in Computer Engineering
Boğaziçi University
2017

ACKNOWLEDGEMENTS

To my love, Elif: because I owe it all to you. Many Thanks!

Foremost, I would like to express my special appreciation and thanks to my supervisor Prof. M. Ufuk Çağlayan, you have been a tremendous mentor for me. I would like to thank you for encouraging my research and for allowing me to grow as a research scientist. Your advices on both research as well as on my career have been priceless. I also would like to express my deepest gratitudes to my co-supervisor Prof. Emin Anarım, for all his support, patience and guidance since my undergraduate studies. My sincere thanks go to Şerif Bahtiyar for not only being such a good friend, but also being a mentor for my PhD studies.

Special thanks to my family. Words cannot express how grateful I am to my parents Ayşenur and Tayfun, and my sister Nihan. Without their patience, encouragement and support, I couldn't complete my thesis. My deepest thanks go to my life-coaches, my late grandfather İhsan Sağnak and my late grandmother Mücella Sağnak. I would like to thank my mother in law Ayşe, my father in law Mustafa and my brothers in law Burak and Ozan for their love and understanding. Also, I would like to express my special appreciation to my aunt Gönül and my uncle Mehmet for their support before and during my PhD studies.

My sincere gratitudes go to Prof. Cem Ersoy, Prof. Fatih Alagöz and Prof. Tuna Tuğcu for providing me an excellent atmosphere in NETLAB. Many thanks also go to my thesis jury members: Prof. Albert Levi, Prof. Semih Bilgen and Assist. Prof. Alptekin Küpçü for their participation in my thesis committee in their rare time and their constructive comments on my thesis. I also would like to thank Haluk Bingöl, Suzan Üsküdarlı and Tunga Güngör for their help and invaluable advices during my PhD studies.

I would like to thank my close friends Kerem, Barış and Serdar together with their family members Nehir, Deniz, Duygu, Esra, Sevim, Tuğrul and Nur for their support, understanding, patience and being always with me when I needed them.

I am grateful to all my colleagues in Boğaziçi Computer Engineering Department. In particular, I owe special thanks to Gürkan Gür, Sezen Bayhan, Can Tunca, Sinan Işık, Cihat Baktır for being such good friends. Moreover, I would like to thank Hande Alemdar, Atay Özgövde, Burak Gürdağ, Akın Günay, Özlem Durmaz İncel, Itır Karaç, Can Kavaklıoğlu, İsmail Arı, Okan Şakar, Halil Ertan, Serhan Daniş, Fadik Kılıçlı, Gökcan Çantalı, Hakan Selvi, Nezihe Pehlivan, Alper Bozkurt, Akif Emre, Kübra Kalkan, Levent Altay, Nadin Kökciyan, Barış Gökçe, Ferit Enişer, Birkan Yılmaz, Salim Eryiğit, Niaz Chalabianloo, Mehran Hosseinzadeh and Can Güven for their great support, encouragement, understanding, discussions and friendship.

Last, but not the least, I would like to thank Serhal Tunçay, Barış Eymen, Burak Can Kurt, Barış Karacan, Burak Erken, Cihan Çaylak, Onur Aktaş, Recep Gökçeli, Önder Türkyilmaz, Ali İ. Aksu, Ersin Ulukan and Serdar Erbatur for their friendship over the years.

This thesis work has been fully supported by the Turkish State Planning Organization (DPT) under the TAM Project, number 2007K120610 and the Scientific and Technical Research Council of Turkey (TÜBİTAK) under BİDEB 2211 A (National Ph.D. Scholarship).

ABSTRACT

NEW DYNAMIC GROUP KEY AGREEMENT PROTOCOLS AND THEIR FORMAL ANALYSIS AND APPLICATIONS

The essence of dynamic group key agreement protocols is to help compute a secure key for a group communication with a dynamic set of participants in distributed systems. Dynamic group key agreement protocols are expected to be used in applications such as conference communications, file sharing systems and mobile ad hoc networks. In dynamic group key agreement protocols, the number of participants may vary in time because of participants are joining or leaving a group. The security of such operations is affected by the existence of backward confidentiality and forward confidentiality, respectively. There are a number of problems related to the use of existing dynamic group key agreement protocols: (i) inefficient fault detection in conference communications, (ii) lack of privacy, violation of availability, inefficient participant revocation, dependency for key escrow in file sharing systems and (iii) insecure cluster head selection in mobile ad hoc networks. In this thesis, we first introduce an improved Dynamic Conference Key Agreement Protocol with dynamic group capabilities to provide better performance in fault detection and correction. Then, we show the application of the proposed protocol on Three-Tier Secure File Sharing System with efficient participant revocation. Second, we propose another Key Agreement Protocol with Partial Backward Confidentiality. The partial backward confidentiality property is a novel security property that allows a new participant to compute the last valid group key just before joining the group. In relation to this protocol, we propose a Private File Sharing System. Third, we propose a Group Key Agreement Protocol for Mobile Ad hoc Networks. In this protocol, we introduce the new concept of secure cluster head selection.

ÖZET

YENİ GRUP ANAHTARI ANLAŞMA PROTOKOLLERİ VE BİÇİMSEL ANALİZ VE UYGULAMALARI

Dağıtık sistemlerde dinamik grup anahtarı anlaşma protokollerinin nedeni dinamik katılımcı kümesinin grup anahtarını güvenli bir şekilde hesaplamasını sağlamaktır. Dinamik grup anahtarı anlaşma protokollerinin konferans iletişimi, dosya paylaşım sistemleri ve mobil tasarsız ağlarda kullanılması beklenir. Dinamik grup anahtarı anlaşma protokollerinde katılımcıların sayısı, katılımcıların gruba katılmaları ve ayrılmaları yüzünden zamanla değişebilir. Gruba katılma ve ayrılma işlemlerinin güvenliği sırasıyla geriye doğru gizlilik ve ileriye doğru gizlilik özellikleri ile sağlanabilir. Halihazırda var olan grup anahtarı oluşturma protokollerinin kullanımında problemlerin listesi şu şekildedir: (i) konferans iletişimde yetersiz hata tespiti, (ii) dosya paylaşım sistemlerinde mahremiyetin sağlanamaması, kullanılabilirliğin ihlali, katılımcıların erişim haklarının yetersiz bir yöntem ile iptal edilmesi, güvenilir üçüncü şahıslara bağımlılık ve (iii) mobil tasarsız ağlarda güvenli olmayan küme başı seçimi. Bu tezde, ilk olarak hata tespiti ve düzeltilmesinde daha iyi performans sağlayan dinamik grup yetenekleri ile iyileştirilmiş Dinamik Konferans Anahtarı Anlaşma Protokolünü öneriyoruz. Daha sonra ise önerilen protokolün etkili erişim haklarının iptal olmasını sağlayan Üç-Aşamalı Güvenli Dosya Paylaşım sistemini öneriyoruz. İkinci olarak, Kısmi Geriye Doğru Gizlilik yöntemini sağlayan Anahtar Anlaşma Protokolünü öneriyoruz. Kısmi Geriye Doğru Gizlilik özelliği, yeni katılımcıya gruba dahil olmadan bir önceki grup anahtarının hesaplanmasını sağlayan yeni bir güvenlik özelliğidir. Önerilen özellik ile bağlantılı olarak, Özel Dosya Paylaşım Sistemini öneriyoruz. Üçüncü olarak, Mobil Tasarsız Ağlar için Grup Anahtarı Anlaşma Protokolünü öneriyoruz. Bu protokolde mobil tasarsız ağlarda yeni ve güvenli küme başı seçimini konseptini öneriyoruz.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS

A	$(n-1)\times(n-1)$ Adjacency Matrix
$\mathcal{A}$	Attribute Set
A_i	Signature for T_i for DCKAP and GKAP-MANET
B_i	A Signature Parameter Generated for T_i in DCKAP and for ω_i in GKAP-MANET
$\mathcal{C}_{comm}(U_i)$	Communication Cost of Participant U_i
$\mathcal{C}_{comm}(\mathcal{U})$	Total Communication Cost of Participants in $\mathcal{U}$
$\mathcal{C}_{comp}(U_i)$	Computational Complexity Cost of Participant U_i
$\mathcal{C}_{comp}(\mathcal{U})$	Total Computational Complexity Cost of Participants in $\mathcal{U}$
$\mathcal{C}_{length}(U_i)$	The Length of Communication Messages for Participant U_i
$\mathcal{C}_{length}(\mathcal{U})$	Total Length of Communication Messages for Participants in $\mathcal{U}$
$\mathcal{C}_{round}(U_i)$	Number of Communication Rounds for Participant U_i
$\mathcal{C}_{round}(\mathcal{U})$	Total Number of Communication Rounds for Participants in $\mathcal{U}$
CK	A Group Key Computed by Using KAP-PBC
$correct(\cdot)$	A KAP-PBC Function for Fault Detection and Correction
$compute(\cdot)$	A KAP-PBC Function for Group Key Computation
$d'_{i,j}$	Secret Key Parameter Generated by U_i for Participant U_j
$dist(U_i,U_j)$	Distance Function for Determining the Number of Hops between U_i and U_j
$e_{i,j}$	A Schnorr Signature for i^{th} Round and j^{th} Participant for KAP-PBC
$\mathcal{F}$	The Set of Shared Files
F_i	An ith Shared File in $\mathcal{F}$
g	Generator for the Subgroup G_q
G_q	Subgroup of Quadratic Residues
$H(\cdot)$	Hash Function
ID_i	Identity of Participant U_i
$invite(\cdot)$	A KAP-PBC Function for Participant Invitation
$join(\cdot)$	A KAP-PBC Function for Joining Participants

$joinPBC(\cdot)$	A KAP-PBC Function for Joining Participants by Applying Partial Backward Confidentiality Property
$\mathcal{K}$	The Set of Group Keys
K_t	Group key computed at time t
$leave(\cdot)$	A KAP-PBC Function for Leaving Participants
L_i	The List of Adjacent Participants for Participant U_i
M	Message
n	The Number of Participants in $\mathcal{U}$
N_{U_i}	The Number of Adjacent Participant in the neighbourhood of U_i
p	A Large Prime Number computed by $p = 2q + 1$
q	A Large Prime Number
$R(x)$	A Random Oracle Model
$rekey(\cdot)$	A KAP-PBC Function for Key Update
$s_{i,j}$	A Schnorr Signature for i^{th} Round and j^{th} Participant for KAP-PBC
$sendPublicKey(\cdot)$	A KAP-PBC Function for Sending Temporary Public Keys
$sendSecretKey(\cdot)$	A KAP-PBC Function for Sending Secret Keys
t_i	Temporary Private Key for Participant U_i
T_i	Temporary Public Key for Participant U_i for DCKAP
T	Timestamp against Replay Attack
T_{exp}	Computational Complexity Cost for Modular Exponentiation Operations
$\mathcal{U}$	The Set of Participants
U_i	An ith participant in $\mathcal{U}$
V	$(n-1) \times (n-1)$ Verification Matrix
$verifyPublicKey(\cdot)$	A KAP-PBC Function for Verifying Temporary Public Keys
$verifySecretKey(\cdot)$	A KAP-PBC Function for Verifying Secret Keys
x_i	Long-Term Private Key
y_i	Long-Term Public Key
Z_p^*	The Set of Integers in base p

α_i	A Signature Parameter generated for $\omega_{i,j}$ for DCKAP and Secret Key Parameter for GKAP-MANET
β_i	A Signature Parameter Generated for α_i in GKAP-MANET
γ_i	A Signature Parameter Generated for $\omega_{i,j}$ for DCKAP and Temporary Public Parameter for GKAP-MANET SCHS Function
δ_i	A Signature Parameter Generated for $\omega_{i,j}$ for DCKAP
θ_i	A Signature for SCHS Function of GKAP-MANET
λ	Security Parameter for CP-ABE
$\omega_{i,j}$	Secret Key of U_i to Share with U_j for DCKAP
ω_i	Temporary Public Key of U_i for KAP-PBC and GKAP-MANET

LIST OF ACRONYMS/ABBREVIATIONS

ABAC	Attribute Based Access Control
ABE	Attribute Based Encryption
ACKA	Auxiliary Conference Key Agreement
Auth	Authentication
CA	Certificate Authority
CDHP	Computational Diffie-Hellman Problem
CKC	Conference Key Computation Function
CP-ABE	Ciphertext Policy Attribute Based Encryption
CRL	Certificate Revocation List
CSP	Cloud Service Provider
CT	Ciphertext
DCKAP	Dynamic Conference Key Agreement Protocol
DDHP	Decisional Diffie-Hellman Problem
DGC	Dynamic Group Capability
DLP	Discrete Logartihm Problem
FDC	Fault Detection and Correction Function
FIBE	Fuzzy Identity Based Encryption
GKA	Group Key Agreement
GKAP-MANET	Group Key Agreement Protocol for Mobile Ad Hoc Networks
HABE	Hierarchical Attribute Based Encryption
HIBE	Hierarchical Identity Based Encryption
ICKAP	Initial Conference Key Agreement Protocol
ID	Identity
KAP-PBC	Key Agreement Protocol with Partial Backward Confidentiality
KP-ABE	Key Policy Attribute Based Encryption
LRE	Lazy Re-Encryption
MA	Malicious Adversary
MANET	Mobile Ad hoc Networks

MK	Master Key
NA	Not Applicable
PFSS	Private File Sharing System
PK	Public Key
PRE	Proxy Re-Encryption
RBAC	Role-Based Access Control
RSA	Rivest Shamir Adleman Public Key Cryptosystem
RSU	Road-Side Unit
SCHS	Secure Cluster Head Selection
SDC	Secret Distribution and Commitment Function
SFSS	Secure File Sharing System
SKCV	Secret Key Computation and Verification Function
SK	Secret Key
SPA	Security against Passive Attacks
SS	Schnorr Signature
TLS	Transport Layer Security
TT-SFSS	Three-Tier Secure File Sharing System
TTP	Trusted Third Party
TPKD	Temporary Public Key Distribution Function
VANET	Vehicular Ad hoc Networks
V2I	Vehicule-to-Infrastructure
V2V	Vehicule-to-Vehicle
WSN	Wireless Sensor Networks
XOR	Logical XOR

1. INTRODUCTION

The societies have been connected more than ever with the evolution of communication technologies, such as the Internet. With rapid developments in Internet, it is possible for participants of a group to involve in group communications. However, providing security in group communications is a significant issue. Cryptographic algorithms and protocols are used to overcome the security issue, for instance encrypting a message to be transferred over insecure communications channels. For message confidentiality by encryption, members of a group have to share a common key, called the group key. Group keys are shared by using key exchange protocols. Key exchange protocols are categorized as key distribution protocols and key agreement protocols. In key distribution protocols, there exists a centralized authority such as a specific member in the group or a trusted third party that distributes the group key. On the other hand, in key agreement protocols, the group key is computed in a distributed manner, by all cooperating group members. The use of key distribution protocols or key agreement protocols depends on the organization of group members in the network. For instance, in decentralized networks, using key agreement protocols are more convenient than key distribution protocols because of difficulties in establishing a hierarchical communication structure. Conversely, if members of the network are hierarchically structured, then key distribution protocols could be a better choice. However, formal modeling of the security for group key agreement protocols is a challenging issue. In this thesis, we focus on distributed and dynamic networks with no hierarchical structure, to propose new protocols with better security and performance properties and also to provide formal analysis and applications.

The earliest key agreement protocol was proposed by Diffie and Hellman in [2], which enables only two participants to agree on a common key. Then, in [3], the secure key exchange of two participants is extended to multi-party secure key exchange by Ingemarsson et al. Following the protocol in [3], various protocols have been published on multi-party setup [3–6]. Moreover, two important group key agreement protocols with and without authentication were proposed by Burmester and Desmedt in [7]. Authentication property, which is one of the well-known security properties of group key agreement protocols, is used for confirming the identities of participants in group communications [8].

Another important property for group key agreement protocols is the fault-tolerance property, which is first introduced by Tzeng in [6]. Fault tolerance is used for preventing true computation of group key by malicious participants in group communication. When a malicious attempt of a participant is detected, the detected entity marked as malicious participant and is removed from the group communication. There have been many studies [9–11] that provide group key agreement protocols with fault tolerance property. In addition, protocol in [12] provides a non-interactive approach for fault-tolerance property to identify and remove malicious participants from the group key computation.

Forward secrecy, which was first proposed in [13], is also an important property for group key agreement protocols. Forward secrecy is used to protect against the compromise of former and subsequent conference-keys of a protocol, if the long-term key of a participant is compromised. The forward secrecy is firstly adopted in [14] for group key agreement protocols. Protocols in [15–17] are also examples for the use of forward secrecy in group key agreement protocols.

The early group key agreement protocols mostly focus on static groups, in which the set of participants does not change until the end of communications session. However, with the growth of technology, dynamic group structures replaced the static groups in multi-party communications. Therefore, group key agreement protocols that were designed for static groups became somewhat outdated since updating the group key resulted in a significant overhead. Consequently, group key agreement protocols have evolved to overcome such overhead by providing dynamic group operations. The most common dynamic group operations are due to the join of new participants and the leave of the existing participants. Such join and leave operations are sometimes called as auxiliary group key agreement operations. In contrast to static group key agreement protocols, to support auxiliary operations, a small subset of participants is selected as an active participant. Active participants, which are the entities in the group, are responsible for updating the group key after any join or leave has occurred. The set of active participants is a smaller group than the original set. This approach is to improve the performance of dynamic group key agreement protocols with join and leave operations.

In addition to the performance improvements in dynamic group operations, providing better security of dynamic group key agreement protocols is also one of the important security goals can not. Dynamic group key agreement protocols have to provide basic security properties such as authentication, fault-tolerance and forward secrecy. In addition, there are two very important properties for the security of dynamic group key agreement protocols, namely backward confidentiality and forward confidentiality [16, 18]. In backward confidentiality, participants who joined the group can not compute former group keys. In forward confidentiality, participants who left the group can not compute subsequent group keys. Backward confidentiality or forward confidentiality are used for providing a fresh key after the group is updated.

1.1. Motivation

Dynamic group key agreement protocols are one of the best candidates for providing security of various multi-party communications applications, where participants are dynamic and they are geographically distributed. Applications of dynamic group key agreement protocols present a challenge for conference communications, secure file sharing systems and Mobile Ad Hoc Networks (MANETs) communications due to the performance and security issues. In this thesis, our main motivation is to propose new protocols with better security and performance properties and also to provide formal analysis and applications.

For conference communications applications, static group key agreement protocols are mostly used since they are for a fixed time period and a fixed set of participants [6,7,14,19]. However, due to the performance degradation in the use of static group key agreement protocols [9,15,17], extracting malicious participants from the conference session is a challenging issue. Dynamic protocols solve this issue by using auxiliary group key agreement operations. In addition, key freshness issue is also important since the absence of backward confidentiality and forward confidentiality properties may cause security vulnerabilities [18]. Therefore, one of our motivations is to provide an efficient fault detection and correction for conference communications by using dynamic group key agreement protocols. Moreover, basic security properties such as forward secrecy and authentication have to be provided in conference communications applications.

For file sharing systems, shared files are usually stored in a cloud server in an encrypted manner by using a shared key. Any participant that has the correct access permissions and the key can retrieve and decrypt the shared file. Access permissions are assigned by central authority in the system. When a participant leaves the file sharing, a responsible authority revokes the access permissions of leaving participant. However, revocation of access permissions is always challenging issue for file sharing systems in terms of performance cost. For instance, to make it unreachable for leaving participant, revocation of access permissions may require re-encryption of a shared file. Since file sharing systems have the same efficiency issues in participant revocation as in conference communications, variant of group key agreement protocols are able to solve this issue, which is an additional motivation in this thesis.

There are also a number of different security problems in using existing dynamic group key agreement protocols in file sharing systems such as the lack of privacy, the violation of availability and the dependency for key escrow. The most important reason of problems for the use of group key agreement protocols in file sharing systems is the existence of backward confidentiality property. Since joining participants can not compute the previous group key just before joining the group, file sharing systems must provide a mechanism to grant access permissions for joining participants. Trusted Third Parties (TTPs) or dedicated participants in the group, for instance Group Managers, are used to overcome this problem. However, if TTPs are involved in file sharing, the privacy of the file is endangered. If dedicated participants distributes group key, there is a possibility for violation of availability due to the single-point-of-failure. Moreover, if TTPs or dedicated participants exist in file sharing systems, a key escrow mechanism provides data recovery keys for encrypted files under certain circumstances [20]. If group key agreement protocols are used in file sharing systems, then there is no need for such recovery mechanism. In this thesis, our additional motivation for file sharing systems is to solve these problems with the provision of partial backward confidentiality in file sharing systems.

Another challenging issue for applications of group key agreement protocols is the communications among participants in MANETs. MANETs are formed by a combination of clusters. Therefore, communications of participants in MANET are categorized as in-cluster

and inter-cluster communications. The first one is the communications of participants that are the member of the same cluster. The second one is the communications of participants that are not the member of the same cluster. In order to organize secure communications for such cluster-based network, most of the existing secure communications protocols use two levels security approach [10, 21, 22]. In the two-level security approach, different group key agreement protocols are needed for in-cluster communications and inter-cluster communications. Cluster heads become the responsible node for decrypting/encrypting the incoming/outgoing messages for inter-cluster communications and in-cluster communications. However, using such approach may affect the security and energy consumption of cluster head. In terms security, availability of cluster members can be violated, because, cluster heads are single-point-of-failure for two-level security approach. In addition, batteries of cluster heads deplete faster since they perform more operations than other participants in their cluster. Dynamic group key agreement protocols can provide a better solution to overcome violation of availability and faster battery depletion problems. Since one group key is used, cluster heads do not need to perform more operations while organizing inter-cluster communications. On the other hand, a secure cluster head selection is another significant issue since existing protocols do not provide such selection in a secure manner. The general approach for cluster head selection is that each participant publicly announces the number of connections with other participants. The one with the maximum number of connection is selected as the cluster head. However, a malicious participant can claim that it has the highest number of connections to be the cluster head. Then, this malicious participant can control all the communications of the cluster. A solution for this problem is to announce the list of the connected participants with the number of connections in a secure manner. Together with the efficient group key computation, secure cluster head selection is our additional motivation for the use of dynamic group key agreement protocols for MANETs.

1.2. Contributions

In this thesis, we propose new protocols with better security and performance properties and also to provide formal modeling an analysis. Major contributions of thesis are as listed below:

(i) For conference meeting applications, we propose Dynamic Conference Key Agreement Protocol with efficient fault correction, called DCKAP [16]. DCKAP uses a modified form of Tseng's protocol [14] as Initial Conference-Key Agreement Protocol (ICKAP) and has new Auxiliary Conference-Key Agreement (ACKA) operations for dynamic group management. DCKAP has better fault correction with respect to Tseng's protocol. Since the security of the Tseng's protocol is preserved and the key freshness is guaranteed with ACKA operations, DCKAP is resistant against known attacks. Moreover, we analyze and compare the scalability of DCKAP with other dynamic group key agreement protocols via simulations for mass join and mass leave operations [23]. Furthermore, we investigate the application areas of DCKAP via simulations. Simulation results show that DCKAP is a scalable group key agreement protocol in terms of participant leaves.

(ii) Therefore, we propose a Three-Tier Secure File Sharing System (TT-SFSS) that use DCKAP for efficiently revoking access permission of leaving participants. Furthermore, TT-SFSS not only present efficient participant revocation but also it provides basic security requirements such as high availability, file privacy, availability and fine-grained access control.

(iii) For secure file sharing systems, we propose the partial backward confidentiality property [24] to overcome security issues in file sharing system depend on the use of group key agreement protocols such as lack of privacy, violation of availability and dependency for key escrow. Partial backward confidentiality, which is a novel security property for dynamic group key agreement protocols, allows one of the joining participants to compute the last valid group key just before joining the group. In addition, we propose an implementation of proposed property in a dynamic group key agreement protocol, namely Key Agreement Protocol with Partial Backward Confidentiality (KAP-PBC). KAP-PBC provides basic security properties of group key agreement protocols such as authentication, fault-tolerance and forward secrecy. Moreover, KAP-PBC is resistant against security attacks designed to endanger the backward confidentiality and forward confidentiality properties.

(iv) In order to show the applicability of KAP-PBC, we propose Private File Sharing System (PFSS) to solve lack of privacy, violation of availability and dependency for key-

escrow problems that appears in the use of existing group key agreement protocols on file sharing systems.

(v) We also propose a Group Key Agreement Protocol for Mobile Ad Hoc Networks, called GKAP-MANET. The proposed protocol is the improved version of the protocol proposed in [17]. One of these improvement is removing the security vulnerabilities given in [18]. In addition, merge operation is added for to increasing the adaptability of the protocol for MANETs. GKAP-MANET contains a new secure cluster head selection mechanism to overcome the malicious attempts of participants to be used for the computation of a new group key. Better performance is provided in terms of reducing the communications and computational cost of group key computation during the execution of the protocol. Furthermore, we present a set of simulations for an example GKAP-MANET application scenario.

Some of contributions of the thesis and additional work during the thesis study have been published in [15, 16, 25–32] or submitted for publication in [23, 24, 33, 34].

1.3. Thesis Outline

The organization of the thesis is as follows:

In Chapter 2, we review the literature with respect to group key agreement protocols, file sharing systems, trust in multi-party communications and context-aware security.

In Chapter 3, DCKAP is presented. In addition, we have presented a comparative study on the scalability of DCKAP and other dynamic group key agreement protocols. Moreover, we have proposed TT-SFSS as an application for DCKAP.

In Chapter 4, we present a new security property, called partial backward confidentiality, with the use of this property for group key agreement protocols as KAP-PBC. In addition, we have proposed secure file sharing systems called PFSS as an application of KAP-PBC.

In Chapter 5, GKAP-MANET is introduced in Chapter 5. Moreover, we have presented simulations for GKAP-MANET on an example network.

Finally, in Chapter 6, our conclusions and future research directions are drawn.

2. LITERATURE REVIEW

In this chapter, we review the literature regarding group key agreement protocols and file sharing systems. First, we define the basic concepts related to group key agreement protocols. Then, we give a detailed review on file sharing systems together with access control, participant revocation and proxy re-encryption.

2.1. An Overview of Group Key Agreement Protocols

In Group key agreement protocol, two or more participants compute a group key by using a common function. There have been many protocols proposed since the Diffie-Hellman Group Key Exchange [2]. One of the most remarkable protocols was proposed by Ingamarsson *et al.* in [3]. The proposed protocol extended the two party group key computation in [2] to multi-party group key computation. Group key agreement protocols are also important cryptographic tools for securely computing a group key in de-centralized and distributed networks since they provide better security and performance on non-hierarchical network structures [19, 35–37]. In this section, we give a detailed overview on performance, security and application areas of group key agreement protocols.

Before presenting the detailed overview of group key agreement protocols, first, we give the following definitions:

Definition 2.1. *Participant, Participant Lists and Participant Types:*

- *Each participant is an entity and denoted as U_i. Moreover, all participants have the same computation power.*
- *The participant in the group or cluster is represented with $\langle U_1, U_2, \ldots, U_n \rangle$, where n is the number of participants in the group.*

- *The list is circular so that $U_{n+i} = U_i$ for some positive $1 \leqslant i \leqslant n$. All participants and the order of the participants are known by each participant.*
- *Participants in a conference session can be categorized in two groups. "Honest participants" are participants that follows the protocol steps and "malicious participant" that try to disrupt the computation of group key or cluster head selection.*

Definition 2.2. Public Parameters. *The protocol uses the following public parameters based on [14] as an extended definition in [38], in which the Decision Diffie-Hellman is believed to be intractable.*

(i) $p = 2q + 1$ *is a safe prime [39], where both p and q are large prime numbers.*
(ii) g *is a generator for subgroup* $G_q = \{i^2 | i \in Z_p^*\}$
(iii) T *is the time-stamp against replay attack.*
(iv) H *is a Hash function.*
(v) ID_i *is the identity of participant* $U_i \in \mathcal{U}$
(vi) V *is an* $(n-1) \times (n-1)$ *verification matrix, is used to determine the result of verifications. Each entry of this matrix is can be either "failure" or "success".*

Definition 2.3. Long-term keys. *Each participant, U_i, has a public - private key pair:*

- $x_i \in Z_q^*$ *is the private key and only the participant* U_i *knows the private key.*
- y_i *is the public key and calculated as* $y_i = g^{x_i} mod\, p$

2.1.1. Performance Analysis for Group Key Agreement Protocol

There exist two criteria that affect the performance of group key agreement protocols, namely communications cost and computational complexity cost. For communications cost, the total number of communications rounds and the length of the transmitted messages by each participant are the key features. For computational complexity costs, computational effort for computing the group key is taken into consideration. Detailed definitions are given in the following sections.

2.1.2. Communications Cost Analysis

The communication cost of a participant is evaluated by the message length and the total number of communications rounds. For any participant $U_i \in \mathcal{U}$, the message length and communication rounds are denoted as $\mathcal{C}_{length}(U_i)$ and $\mathcal{C}_{round}(U_i)$, respectively. The message length corresponds to the total length of parameters in a messages broadcasted by participant U_i. In addition, the total number of communication rounds is used for representing the total number of messages exchanged by U_i while calculating the group key. The total communications cost of a participant is calculated as follows:

$$\mathcal{C}_{comm}(U_i) = \mathcal{C}_{length}(U_i) + \mathcal{C}_{round}(U_i) \tag{2.1}$$

Communications cost analysis is also used for measuring the total number of communications performed by all participants in group. For instance, let $\mathcal{U} = \{U_1, U_2, U_3, \ldots, U_n\}$ be the set of participants in the group, the total message length and communications rounds of participants in the group during the execution of the group key is denoted as $\mathcal{C}_{length}(\mathcal{U})$ and $\mathcal{C}_{round}(\mathcal{U})$, respectively. The total communications cost of the group is calculated as follows:

$$\mathcal{C}_{comm}(\mathcal{U}) = \mathcal{C}_{length}(\mathcal{U}) + \mathcal{C}_{round}(\mathcal{U}) \tag{2.2}$$

2.1.3. Computational Complexity Cost Analysis

Another important performance criteria for group key agreement protocols is the computational complexity cost. During the computation of a group key, each participant makes multiplications, summations, logical XORs and exponentiation in modular base. The most time-consuming one among these operations is the modular exponentiation. Therefore, we only concentrate on modular exponentiations for the computational complexity cost analysis of group key agreement protocols. Throughout the thesis we use $T_{exp} = O(x^y mod z)$ notation for modular exponentiations. In addition, the computation cost is denoted as $\mathcal{C}_{comp}(U_i)$ for a participant U_i. Moreover, the computational complexity cost of group during the computation of group key is denoted as $\mathcal{C}_{comp}(\mathcal{U})$ for the set of participants $\mathcal{U} = \{U_1, U_2, U_3, \ldots, U_n\}$.

2.1.4. Security of Group Key Agreement Protocols

In this section, first, we introduce the mathematical background and basic properties for the security of group key agreement protocols. Next, we define the security properties of dynamic group key agreement protocols.

2.1.4.1. Background. The first definition for the mathematical background of group key agreement protocols is that the Discrete Logarithm Problem (DLP). DLP use the one-wayness property [40] of exponentiation of certain cyclic groups under multiplication [41]. According to the one-wayness property: it is easy to compute outputs for every inputs but computationally hard to invert input from given outputs. DLP is used as a base for most of public key cryptosystems such as RSA [42] and ElGamal [43]. The formal definition of DLP is as follows:

Definition 2.4. *Discrete Logarithm Problem. Let g be a generator of some cyclic group G_p, where p is a large prime, for instance a safe prime. For any given x, g, p, calculating $y \equiv g^x mod p$ is easy but calculating x from given y, g, p is computationally hard.*

The second definition in this section is the definition of Computational Diffie-Hellman Problem (CDHP). CDHP is defined in [44] to prove that breaking Diffie-Hellman protocol is as hard as DLP. The formal defintion of CDHP is as follows:

Definition 2.5. *Computational Diffie Hellman Problem. Let g be a generator of some cyclic group G_p, where p is a large prime. For integers a and $b \in Z_p$, it is hard to compute g^{ab} from given g, g^a, g^b.*

The third definition for the security is Decisional Diffie-Hellman Problem. It was shown that yet alone CDHP is insufficient proof for using Diffie-Hellman as cryptographic purposes by Dan Boneh in [38]. Therefore, Boneh defined more stronger definition for key exchange protocols, namely Decisional Diffie-Hellman Problem (DDHP). The formal definition of DDHP is as follows:

Definition 2.6. *Decisional Diffie-Hellman Problem. Let g be a generator of some cyclic group G_p, where p is a large prime. The following two message transcripts are computationally indistinguishable:*

- $\langle g^a, g^b, g^{ab} \rangle$*, where a and b are integers randomly and independently selected from Z_p.*
- $\langle g^a, g^b, g^c \rangle$*, where a, b and c are integers randomly and independently selected from Z_p.*

The fourth and final definition related to group key agreement security is Random Oracle Models. Random oracle models are important for interconnecting the implementation and formal representation of cryptographic protocols [45]. In cryptography, random oracle models are used as a black box function that randomly map outputs for given inputs. Random oracle can be formally represented as given in the following definition:

Definition 2.7. *Random Oracles. Let $R : \{0,1\}^* \longrightarrow \{0,1\}^\infty$ be a random oracle used for selecting each bit of $R(x)$ uniformly and independently for a given x.*

2.1.4.2. Basic Security Properties. Basic security properties of group key agreement protocols are known as authentication, completeness, fault-tolerance and forward secrecy. In group key computation, authentication property is used for the identity confirmation of participants. In key agreement protocols, authentication can be satisfied by either certificates or identities (IDs) [46]. Protocols that provides authentication by using certificate-based or identity-based controls are called as authenticated protocols in [7, 9, 15, 16, 19, 36, 47]. If a protocol does not satisfy the authentication property, then it is called an unauthenticated protocol [7, 48, 49]. Authenticated protocols are more advantageous than unauthenticated ones since they are able to operate unauthenticated networks. On the other hand, it is possible to convert unauthenticated protocols into authenticated protocols such as given in Ateniese *et al.* [50], in which unauthenticated Diffie-Hellman-variants in [51] were converted into authentication property. However, protocols in [50] suffer from the lack of formal security model. The first formal security models for authentication property was proposed in [52, 53] by Bresson *et al.* Unauthenticated group key agreement protocols are out of scope for this thesis.

Completeness property is necessary to show if the correct group key is computed by all of the participants in the group if they fully follows the steps of a protocol as defined in [6]. Another remarkable studies that use completeness as a security property are proposed in [16, 17].

The fault-tolerance property, which is introduced by Tzeng in [6], is necessary for detecting and correcting the malicious behavior of participants during key computations. Other group key agreement protocols with fault tolerance property are in [9, 10, 12, 14, 15]. For instance, protocols in [9, 10, 15] improved the group key computation performance with respect to Tzeng's protocol. The protocol in [12] provides non-interactive approach for fault-tolerance property to identify and remove malicious participants from the group key computation. In addition to the fault-tolerance property, the forward secrecy property is also crucial for providing security against compromising group keys if the long-term private key of any participant is compromised [14]. The following cases are considered as faults in group key computation:

- A malicious participant can cheat the honest participants by sending wrong key values, and
- A malicious participant can cheat the honest participants by identifying an honest participant as a possible malicious participant.

The final security property that we present in this section is forward secrecy property. Forward secrecy was proposed in [13] and adopted to the conference-key agreement protocols in [14] as an extended protocol of [6]. Forward secrecy, which is used to protect against the compromise of former and subsequent group keys of a protocol, if the long-term key of a participant is compromised, is also defined as a standard in Eliptic Curve Cryptography (ECC) [54] and key agreement and key transport using ECC [55]. The formal definition of forward secrecy is given in the following definition:

Definition 2.8. *Forward Secrecy. Assume that $\mathcal{K} = \{K_1, K_2, K_3, \ldots, K_n\}$ is set of n different group keys that computed between t_0 and t_{n-1} time interval. Let x_i and y_i be the private and public key pair of any participant $U_i \in \mathcal{U}$ that are used during the generation of all keys*

in $\mathcal{K}$, respectively. For any group key agreement protocol that works on the given setup, if compromise of x_i does not results in the compromise of all keys in $\mathcal{K}$, then the protocol provides forward secrecy.

2.1.4.3. Security of Dynamic Groups Key Agreement Protocols. In dynamic group key agreement protocols, participants are allowed to join the group or leave the group after group key is computed. However, such dynamic group operations may cause some security vulnerabilities for group key computation. For the sake of secure group communications, backward confidentiality and forward confidentiality properties [16] have to be provided. In backward confidentiality, previous group keys cannot be obtained by the participants who join the group and this property is formally defined as follows:

Definition 2.9. *Backward Confidentiality. Let U_i be the new participant that will join into the group communication in $\mathcal{U}$. Let K_t and K_{t+1} be the group keys of $\mathcal{U}$ before and after U_i joins into the group, respectively. Any group key agreement protocol satisfies backward confidentiality property if $K_t \neq K_{t+1}$ and K_t cannot be obtained by using $\mathcal{K}'$ [16].*

In forward confidentiality, subsequent group keys cannot be obtained by the participants who left the group. The formal definition of forward confidentiality is as follows:

Definition 2.10. *Forward Confidentiality. Let $U_i \in \mathcal{U}$ be the leaving participant. Let K_t and K'_{t+1} be the group keys of $\mathcal{U}$ before and after U_i leaves the group, respectively. Any group key agreement protocol satisfies forward confidentiality property if $K_t \neq K_{t+1}$ and K_{t+1} cannot be obtained by using $\mathcal{K}$ [16].*

2.1.5. Applications of Group Key Agreement Protocols

Group key agreement protocols have many application areas such as conference communications, ad hoc network communications, wireless sensor networks, etc. In this section, we address literature review for applications of group key agreement protocols with respect to the use in different networks.

2.1.5.1. Conference Communications. One of the most significant usage of group key agreement protocols is the conference communication. Group key agreement protocols are also defined as conference key agreement protocols when they are used for computing group key for secure conference communication. One of the most remarkable study for such communication systems was presented by Burmester and Desmedt in [7]. Two important group key agreement protocols with and without authentication were proposed by in this study. The non-authenticated protocol in [7], namely BD protocol, is known as one of the most efficient static conference key agreement protocol in literature. There exist various protocols, which were designed on the mathematical base of BD protocol such as protocols in [17, 33].

Another important conference key agreement protocol was proposed by Tzeng in [6]. Tzeng's protocol is the first protocol that uses fault-tolerance property in group key agreement protocols. Fault tolerance is used for detecting and excluding the malicious participants from conference key computation. In [4], a conference key agreement based on secret sharing was proposed. In the proposed protocol, entity authentication is replaced with group authentication by using secret sharing. In addition, protocol proposed in [14] is also important since the forward secrecy property was firstly used for conference key agreement protocols. Forward secrecy is necessary to protect against the compromise of former and subsequent conference-keys of a protocol, if the long-term key of a participant is compromised. In [9], Huang *et al.* proposed another conference key agreement protocol with fault tolerance property. Huang *et al.* updated the key computation functions in [6] to have better performance by using less modular exponentiation operations. In addition, protocol in [15] updated the Huang *et al.* protocol with the forward secrecy property. Also, conference key agreement protocol in [16] provides more efficient fault correction by using dynamic group operations.

2.1.5.2. Ad hoc Networks. Another important application area for group key agreement protocols is Vehicular Ad hoc Networks (VANETs). There are two types of communication in VANETs, Vehicle-to-Vehicle (V2V) and Vehicle-to-Infrastructure (V2I). The former one is the communication among vehicles. The latter one is the communication of vehicles with static nodes that are located beside the streets or highways. Such infrastructures in

VANETs are sometimes called as Road-Side Unit (RSU). RSUs are responsible for providing necessary information for V2V communication. One of the important applications of key agreement protocols for VANETs is proposed in [56]. They showed a two-layer structure as vehicle-to-infrastructure (V2I) and vehicle-to-vehicle (V2V). In V2I, communication vehicles obtain necessary information by querying RSUs to communicate with other vehicles. For V2V communication, Diffie-Hellmann based approach is used for computing common key among two vehicles.

In [57], a probabilistic key distribution protocol was proposed for secure communication. In this approach, instead of returning a single key for current VANET, the RSU returns key pool with k keys. As long as they share the same keys, the two vehicles can securely connect to each other. In [58], a beaconing approach with key agreement in VANETs was proposed. A beaconing is one of the communication modes that detects other vehicles in the neighbourhood with periodic sending and listening to the beacon packets. After the neighbour node is detected, the secret keys are agreement to generate a symmetric signature. In [59], the proposed protocol is based on the group signature by using mutual identification and key agreement scheme in secure VANETs. The protocol contains the following steps: setup, join and key agreement. The RSA public key encryption is used throughout the protocol. The group manager computes and distributes the secret and public keys. Then, these keys are used to generate group private key. For participants joins, Diffie-Hellman key agreement, mutual identification and verification schemes are used.

In addition to the VANETs, group key agreement protocols are also applied to Mobile Ad hoc Networks (MANETs). A comprehensive surveys regarding applications of the group key management protocols for network layer security in ad hoc networks are given in [60] and categorization of key management schemes based on contributory key based, public key based and symmetric key based management schemes are given in [61]. Energy efficiency is also important concern for MANETs because of the fast battery depletion problem of mobile devices. Protocols in [62, 63] studies group key agreement protocols with respect to the energy efficiency point of view.

One of the most important examples for the use of group key agreement protocols in MANETs is the protocol in [51]. Proposed protocol uses Diffie-Hellman key exchange protocol to handle group key agreement for dynamic group operations. Protocol in [64] uses the protocol in [51] to provide admission control.

2.1.5.3. Wireless Sensor Networks. The last important application area that we introduce in this thesis is group key agreement protocols is the Wireless Sensor Networks (WSNs). WSNs consist of devices with restricted computation powers. In order to avoid the computational cost overhead due to the exponentiation in group key computation, the main aim for establishing secure key among sensor nodes of a WSN is to use key pre-distribution schemes [65]. In key pre-distribution schemes, a random key pool which consists of possible peer-to-peer keys are selected. Then, a smaller key subset from a random key pool is selected and deployed into a sensor node. When the sensor is activated, it tries to find the common keys together with other sensors in its neighbourhood. Key establishment is achieved once a common key is found in key pools of two different sensor nodes. Next, the confidentiality of communication among these two sensors is provided by using the common key. The main drawback of key pre-distribution scheme is that two sensor nodes may not found the common key in their key subsets, which also cause disconnected nodes in the network.

One approach to avoid disconnected nodes in the network is to increase the probability of finding common keys in any two neighbour sensor nodes [66]. The proposed protocol use two different key pools for mobile sink and pairwise key establishment among sensor nodes to increase connectivity.

Although key pre-distribution looks more convenient for key establishment in sensor networks, today's sensor devices are more capable in term of computation power than the ones in twenty years ago. In [32] and [28], implementations of group key agreement protocol in [7] on Arduino Fio and ATMEGA 2560 were proposed. According to the test results in real-time environment, group key agreement protocol in [7] provide better performance than key pre-distribution scheme in [65]. With the increased popularity of Internet of Things (IoT)

systems, we expect to see the emerging implementations of group key agreement protocols on wireless sensors.

2.2. An Overview of Secure File Sharing Systems

Cloud storage services have obtained an irreplaceable place in today's world. As such services are being developed rapidly, plenty of companies started to outsource their important data in cloud servers. Although using cloud storage seems quite profitable regarding the low maintenance cost of the servers, it also brings along some security threats. These threats can be categorized as internal and external threats. In the former one, the Cloud Service Provider (CSP) cannot be fully trusted, as they might want to learn the content of the shared data. Therefore, the privacy of shared data is endangered. The latter one arises when the cloud is not securely protected against attempts of malicious participants. Such participants try to access the shared data illegally or to corrupt shared content. Therefore, the existence of security threats caused by storing data in a cloud has encouraged researchers to develop secure file sharing systems. The aim of this section is to provide a detailed overview of secure file sharing systems regarding the provided protection against internal and external threads. First, we review the important secure file sharing systems in literature. Then, we analyze the SFSSs with respect to the following criteria:

(i) Access Control: Regulating the permissions to access the encrypted shared data in a file server.
(ii) Participant Revocation: Revoking the operational power of a participant on shared data.
(iii) Proxy Re-Encryption (PRE): Delegation of executing cryptographic operations to update the encrypted text from participants to the file server without allowing to access the plaintext.

Security and performance issues of file sharing systems are not restricted to access control, participant revocation and proxy re-encryption, but also, dynamic groups [67], key management, anonymity [68], extended searching functionality for encrypted data on cloud

server [69], assured deletion of shared files [70] and storing shared data on distributed servers [71] are important properties of secure file sharing systems.

Managements of dynamic groups are another challenging issue for file sharing systems. File sharing system in [67] is a good example for handling of dynamic groups. In the proposed protocol, users of the proposed system are categorized as group managers and group members. Group managers are responsible for generating the system parameters, providing user registration and revoking access permissions of malicious or leaving users. Group members are the set of registered users that store their own data into the cloud and share them with other users in the system. Existence of file confidentiality keys allows participants to communicate with file servers over insecure channels. In addition, proposed system provides access control mechanism for managing the access permissions of authenticated and non-authenticated users. In [68], Mona, secure multi-owner data sharing system was proposed. Mona also provides mechanism to handle of dynamic groups in the cloud environment. The most important novelty in Mona is that it efficiently operates on dynamic groups for granting immediate access to shared files for joining participants and revocation of access permissions for leaving participants. In addition to the dynamic groups, Mona allows more than one participants to own a shared file. Moreover, the proposed system provides identity privacy to allow participants to operate and share files anonymously.

In [69], a practical and secure thin model data sharing scheme was proposed. In the proposed scheme, users are able to encrypt and share data without revealing the shared confidentiality keys, it is flexible and easy to download and decrypt the shared data by using only one master key and it is allowed to search a keyword in the shared data.

Assured deletion is another important property for file sharing systems as proposed in [70]. According to the assured deletion, when a shared data is deleted in the cloud server, no user, including the owner of the file, is able to access the data. In [70], a file sharing system called FADE was presented to show how to apply assured deletion in file sharing systems. Moreover, FADE provides fine grained access control together with fault-tolerant key management.

File sharing systems may divide and store shared data in multiple storages servers. In order to get the full content, a user need to access these storages simultaneously. In [71], an authenticated key exchange protocols for network file systems that provides parallel access to multiple storages was proposed. In the study, three different protocols were proposed called pNFS-AKE-I, pNFS-AKE-II and pNFS-AKE-III in order to improve the limitations of Kerberos [72] such as scalability on parallel access to multiple storage, missing forward secrecy property and escrow-freeness. In pNFS-AKE-I, a modified version of Kerberos with scalability on parallel access was proposed. In addition to the scalability on parallel access, pNFS-AKE-II provides partial forward secrecy. Finally, pNFS-AKE-III was designed to overcome almost all of the limitations of Kerberos with full forward secrecy.

2.2.1. Access Control Mechanisms

Access control mechanisms can be categorized as coarse-grained access control mechanism an fine-grained access control mechanism. In file sharing systems, coarse grained access control mechanisms mostly use Role-Based Access Control (RBAC) [73], where access permissions are granted to a role. In fine-grained access control, access permissions are granted by the use of some policies together with the use of attributes. Therefore, fine grained-access control mechanisms provide more flexible access control definitions than coarse-grained access control.

For fine grained access control mechanisms, Attribute Based Access Control (ABAC) models are used as defined in [74, 75]. The most well-known application of ABAC models in literature are the Attribute-Based Encryption (ABE) scheme, which was firstly introduced by Sahai and Waters in [76]. ABE was proposed as a variant of Identity-Based Encryption (IBE) scheme called Fuzzy Identity-Based Encryption (FIBE). According to the FIBE, only the participants, whose private keys have at least an arbitrary number k of overlapping attributes with ciphertext can decrypt the encrypted data, where $1 \leqslant k \leqslant n$ and n is the length of the attribute set. Even though FIBE provides error-tolerance and security against collision attacks, the expressiveness of the scheme is limited. Therefore, FIBE cannot be applied to large systems, where the access control should be more fine-grained.

Goyal *et al.* suggest a variant of ABE system for fine-grained access control in [77]. The proposed ABE system is called Key-Policy ABE (KP-ABE). In KP-ABE, ciphertexts are labeled with set of attributes and private keys of users are associated with access structures that decide which ciphertexts the user is allowed to decrypt. Unlike FIBE, which prioritizes error-tolerance, the KP-ABE scheme focuses on access control. Therefore, a finer-grained access control with respect to the classical ABE schemes is achieved in the study.

The complementary of KP-ABE is called Ciphertext-Policy ABE (CP-ABE). Although CP-ABE is presented as an idea in [77], no construction of such scheme is given in the study. The first realization of a CP-ABE system is implemented by Bethencourt *et al.* in [78]. CP-ABE uses the opposite logic of KP-ABE. In CP-ABE, a private-key of a user is associated with a set of attributes and a ciphertext, which specifies an access policy over a defined set of attributes for the system. The main difference between KP-ABE and CP-ABE is that in KP-ABE, the encryptor has no control over who can access the ciphertext. Instead, the encryptor must trust the key issuer for distributing appropriate keys to appropriate users. Therefore, in CP-ABE, the encryptor decides who should have the access to the encrypted data by using an appropriate access structure for the ciphertext. Bethencourt *et al.* proves the security of the scheme in [78] under the generic group model. However, a security proof under the standard model is not provided. The difference between the two groups is as follows: In the generic group model, the adversary can only access a randomly chosen encoding of a group, instead of an efficient encoding. On the other hand, in the standard model, there does not exist such limitation. As opposed to the generic group model, standard model limits the adversary only by computational power and time. Waters provides an improved version of the CP-ABE scheme in [1]. The CP-ABE scheme realized in this study is proved to be at least as efficient as the one in [78]. However, in [1], the scheme is also proved to be secure under the standard model, making it more applicable for real-world scenarios.

Definition 2.11. *Ciphertext-Policy Attribute Based Encryption (CP-ABE). Let* $\mathcal{A} = \{A_1, A_2, \ldots, A_N\}$ *be the set of attributes for set of participants* $\mathcal{U} = \{U_1, U_2, \ldots, U_N\}$*, where* $A_i \in \mathcal{U}$ *is the attribute associated with* $U_i \in \mathcal{U}$*. Algorithm of CP-ABE operates as in Figure 2.1.*

1 **Setup($\lambda, \mathcal{A}'$):** The setup algorithm generates public key PK and a master key MK by using the security parameter λ and the set of attributes $\mathcal{A}' \subseteq \mathcal{A}$.

2 **Encrypt($PK, M, \mathbb{A}'$):** The algorithm encrypts the message M, by using the access structure that consists of the attributes in $\mathcal{A}'$ and the public key PK to generate ciphertext CT. A participant U_i can decrypt the CT if and only if A_i is in $\mathcal{A}'$.

3 **Key Generation($MK, \mathcal{A}'$):** This algorithm generates the secret key SK by using the master key MK and the set of attributes $\mathcal{A}'$.

4 **Decrypt(PK, CT, SK):** The algorithm decrypts the ciphertext CT by using the public key PK and the secret key SK. If SK associated with $\mathcal{A}'$ satisfies the access structure $\mathbb{A}$ then, the algorithm decrypts and returns the message M.

Figure 2.1. Ciphertext-Policy Attribute Based Encryption [1]

Introducing access control through ABE raises significant performance issues. One such issue is the difficulty in user revocation. The initial design of ABE schemes usually does not support an efficient way of revoking users. In general, the amount of time it requires to revoke the access permissions of a user depend on either the number of revoked users or the number of total users in the system. Therefore, SFSS that uses ABE to achieve fine-grained access control should be modified to handle scalable user revocations such as [74] and [75].

Another example for efficient participant revocation is in [79], which proposes Hierarchical Attribute-Based Encryption (HABE) and scalable user revocation for sharing data in cloud servers. HABE is the composition of Hierarchical Identity-Based Encryption (HIBE) and Ciphertext Policy Attribute Based Encryption (CP-ABE). The proposed system uses the advantages of hierarchically generation of keys in HIBE and the flexible access control of CP-ABE. In order to provide the scalable revocation, the proposed system also uses proxy re-encryption and lazy re-encryption.

Scalable and secure sharing of personal health records in cloud computing using attribute based encryption in [74] is the another important example for the use of ABE in file sharing systems. By using attribute based encryption, the proposed system satisfy the

following requirements:

- Data confidentiality to protect the patient data from the access attempts of unauthorized users.
- On-demand revocation for the users whose attributes are no longer valid.
- Write access control against the attempts of unauthorized users.
- Scalable and efficient in terms of complexity costs of key management, communications between the server and the users.

In addition to ABAC and RBAC, a group signature can be used as access control mechanism [80]. The proposed system provides authentication and data privacy within reasonable time and with reasonable amount of data. Moreover, the data traffic in cloud traffic is reduced in order to increase QoS. Furthermore, the system also provides group signature to be able to use in access control. Such access control mechanism is called as dual grand access control mechanism. Finally, user anonymity and traceability is protected by using identity privacy, in which the cloud service provider should not be able to distinguish the identity of participant that signs the shared data.

2.2.2. Proxy Re-Encryption

Converting a ciphertext encrypted by one key to a ciphertext encrypted for another key by using an untrusted proxy server is called proxy re-encryption (PRE) [81]. In PRE, parties publish a proxy key for allowing the untrusted proxy server to alter the ciphertext without viewing the plaintext. Formal definition of PRE is as follows:

Definition 2.12. *Proxy Re-Encryption (PRE). Let CT be the ciphertext of message M encrypted by using the key k_a. In PRE, a proxy server is allowed to convert CT with $k_{a\leftrightarrow b}$ to obtain a ciphertext CT' which is encrypted version of M by using k_b for $k_a \neq k_b$ [81].*

PRE schemes are mostly used in PKE schemes [81–83]. Therefore, any ciphertext generated by using PKE such as the identity of a participant [83], shared files [75, 79, 84] or e-mail content [85] can be transformed. On the other hand, there exist various studies on the

security of PRE such as the security against Chosen Ciphertext Attacks [83, 86].

In this thesis, we mostly concentrate on the applications of PRE on secure file sharing systems [75, 79, 84, 87]. In [87], file sharing system called CloudSeal was proposed. The proposed system allows any participant in the group to securely sharing a content with other participants via multicasting. In [84], the proposed system uses PRE to manage the access control without granting the full permission in a centralized access control server. The study in [79] proposes Hierarchical Attribute-Based Encryption (HABE) and scalable user revocation for sharing data in cloud servers. HABE uses PRE for updating the file confidentiality key on cloud server when a user's access permissions are revoked. In [75], a time-based proxy re-encryption scheme for secure data sharing in cloud environment (TimePRE) was proposed. By using proxy re-encryption, TimePRE provides efficient user revocation since the file confidentiality key can be updated without the need of an online data owner.

2.2.3. Participant Revocation

Participant revocation is one of the most important properties of file sharing systems. When a participant leaves the file sharing, he/she could not view or update the shared file. Therefore, a secure file sharing system has to provide a mechanism for revoking the access permission of the leaving participants. Such operation is called participant revocation. A participant in file sharing can be removed from the file sharing due to his/her malicious behaviour such as updating the shared file by some file or disrupting the communications in file sharing group. Therefore, owner of malicious behaviour is removed from the file sharing group and permissions for accessing the shared file is revoked from the malicious participant. In order to mark participants as an untrusted participants for future file sharing operations, malicious participants are stored in a list called revocation list as Certificate Revocation List (CRL) [88].

Revoked users should not be allowed to access the cloud as their rights have been denied. Therefore, the file confidentiality keys and sometimes the user keys need to be updated each time a user left the group. If this update process is not scalable and gets affected by the number of total users and the number of revoked users, serious computation overhead and

performance degradations occur in the system. In general, there are two revocation approach for updating the file access permissions in file systems. One of these approaches is the active revocation [89]. In active revocation, if access permissions of a participant is revoked, then the all necessary information that can be used by the revoked participant is updated immediately. However, the cost of updating the access information of a shared file is not cost-efficient in terms of computation of necessary cryptographic operations. Therefore, the second approach Lazy Re-Encryption (LRE) or lazy revocation [89] preferred for revoking the access permissions of leaving participants. In LRE, the update of access permissions is postponed to the first update of the shared file after the revocation of participants [90, 91]. LRE is mostly used with PRE as given in [75, 79] to improve the performance and scalability in the revocation process. A formal definition of LRE is as given below:

Definition 2.13. *Lazy Re-Encryption (LRE). Let $\mathcal{U} = \{U_1, U_2, \ldots, U_N\}$ be the set of participants and let $\mathcal{F} = \{F_1, F_2, \ldots, F_N\}$ be the set of shared files by the participants in $\mathcal{U}$. In LRE, when a participant U_i is revoked, re-encryption of files in $\mathcal{F}$ with updated key is delayed until the update of each file $F_i \in \mathcal{F}$ because of performance issues [89].*

2.3. Discussions

In this chapter, we have reviewed concepts related to group key agreement protocols and file sharing systems to increase the understanding the concepts proposed in thesis. First, we have presented background information for group key agreement protocols in terms of performance analysis and security analysis. Then, we have addressed the application areas of group key agreement protocols. Finally, a detailed overview of secure file sharing systems with respect to access control, proxy re-encryption and lazy re-encryption has been given.

3. DCKAP: DYNAMIC CONFERENCE KEY AGREEMENT PROTOCOL

The societies have been connected more than ever with the evolution of communication technologies, such as the Internet. It is possible for participants of a group to organize a conference meeting without assembling together with the rapid developments of the Internet. However, providing the security for the communications of the conference is a challenging issue. Cryptographic algorithms are used to overcome the issue. These algorithms are useful when a common key among participants is fresh. The freshness is accomplished by using conference-key establishment protocols to generate or update the key for each session. Key establishment protocols are categorized as key distribution and key agreement. Key distribution protocols may need a centralized authority, such as member in the network or trusted third party, to distribute conference-key to participants. Key agreement protocols enable all participants in the meeting to compute a common key. These protocols are preferred with respect to the type of network and the duration of the communications. For instance, in the conference networks, short-term keys are used for each session of decentralized and distributed networks. Therefore, key agreement protocols are more suitable for conference networks.

In general, static conference key agreement protocols are mostly used for conference communications since conferences are arranged for a fixed time period and a fixed set of participants [6, 7, 14, 19]. However, due to the performance degradation in the use of static group key agreement protocols [9, 15, 17], detecting and extracting malicious participants from the conference session is a challenging issue. In dynamic conference key agreement protocols, this issue is solved by using auxiliary group key agreement operations. In addition, key freshness issue is also challenging since absence of backward confidentiality and forward confidentiality properties may cause security vulnerabilities [18]. Therefore, one of our motivations for this chapter is to provide a efficient fault detection and correction for conference communications by using dynamic group key agreement protocols. Moreover, basic security properties such as forward secrecy and authentication has to be provided in

conference communications applications. Furthermore, analyzing the scalability of DCKAP and dynamic group key agreement protocols is our another motivation for this chapter since.

Contributions of the chapter are as follows:

(i) We propose Dynamic Conference Key Agreement Protocol (DCKAP) [16]. DCKAP uses a modified form of Tseng's protocol [14] as Initial Conference-Key Agreement Protocol (ICKAP)

(ii) DCKAP consists of dynamic group operations, namely Auxiliary Conference Key Agreement (ACKA) Operations for efficiently updating the group key when the conference group is updated. By using ACKA operations, we improve the performance fault correction functionality with respect to the previously proposed conference key agreement protocols.

(iii) We analyze and compare the scalability of DCKAP with other dynamic group key agreement protocols via simulations for mass join and mass leave operations. Simulation results show that DCKAP is a scalable group key agreement protocol in terms of participant leaves.

(iv) We investigate the application areas of DCKAP based on the simulation results. Since DCKAP provides better for participant leaves. We have decided to use DCKAP to revoke access permission of participants in file sharing systems. Therefore, we propose a Three-Tier Secure File Sharing System (TT-SFSS) that use DCKAP for efficiently revoking access permission of leaving participants. TT-SFSS not only present efficient participant revocation but also it provides basic security requirements such as file privacy, availability and fine-grained access control.

The organization of the chapter is as follows. In the next section, we give the comparison of dynamic conference key agreement protocols with respect to security and performance criteria. In Section 3.2, we introduce DCKAP. Security and performance analysis are in Section 3.3 and 3.4, respectively. In Section 3.5, scalability analysis of DCKAP and other dynamic group key agreement protocols is presented. Section 3.6 proposes TT-SFSS. Finally, discussions and concluding remarks are given in Section 3.7.

3.1. Comparison of Conference Key Agreement Protocols with respect to Security and Performance Criteria

In this section, we compare DCKAP with previously proposed conference key agreement protocols literature as shown in Table 3.1. The criteria that are used in the comparison are listed as follows:

(i) Authentication: If the protocol contains a mechanism to detect whether the participant is a member of the conference or not, then this protocol is called an authenticated protocol. Otherwise, it is called as a non-authenticated protocol.

(ii) Fault-Tolerance: This property is used to detect the faulty broadcast messages during the communication of participants. If any fault is detected, then the owner of the message is marked as a possible malicious participant. In the fault correction function, messages of possible malicious participants are re-verified. Then, malicious participants are excluded from the set of participants.

(iii) Forward Secrecy: This property is used to protect the conference-key against compromises of produced conference-keys, such as the compromise of a long-term private key of a participant.

(iv) Dynamic Settings: The protocols are compared according to the dynamic group operations. The numbers given for each protocol are the total number of operations per protocol. There exist four dynamic group operations in the literature, single or mass join, single or mass leave, merging groups, dividing groups into sub-groups.

(v) Key Freshness: whether the computed key is fresh or not.

Table 3.1. Comparison of Protocol Properties

Protocols	**Authenticated**	**Fault-Tolerance**	**Forward Secrecy**	**Dynamic Operations**	**Key Freshness**
Burmester and Desmedt (1994) [7]	✓	✗	✗	NA	✗
Steiner *et al.* (1996) [92]	✗	✗	✗	NA	✗
Li and Pieprzyk (1999) [4]	✓	✓	✗	NA	✓
Tzeng and Tzeng (2000) [35]	✓	✓	✗	NA	✓
Foss (2000) [93]	✓	✗	✗	2	✓
Horng (2001) [49]	✗	✗	✓	NA	✓
Tzeng (2002) [6]	✓	✓	✗	NA	✓
Boyd and Nieto (2003) [94]	✓	✓	✗	NA	✓
Shi *et al.* (2004) [11]	✓	Partial	✗	1	✓
Tseng (2005) [14]	✓	✓	✓	NA	✓
Steiner *et al.* (2005) [48]	✓	✗	✗	NA	✗
Tseng (2007) [17]	✓	✓	✓	2	✗
Chang *et al.* (2007) [95]	✗	Partial	✗	NA	Partial
Katz and Yung (2007) [47]	✓	✗	✓	NA	✓
Huang *et al.* (2009) [9]	✓	✓	✗	NA	✓
Wu *et al.* (2009) [96]	✓	✓	✗	NA	✓
Wang (2012) [97]	✓	✓	✗	NA	✓
Zhao *et al.* [10]	✓	✓	✓	NA	✓
Cheng *et al.* (2013) [98]	✓	✓	✗	2	✓
Chung (2013) [5]	✓	✗	✓	2	✓
DCKAP	✓	✓	✓	4	✓

As shown in Table 3.1, we have compared DCKAP with other protocols in the literature. According to comparisons, only DCKAP provides all of the basic security properties of conference-key agreement protocols with dynamic group operations. On the other hand, some of the protocols given in Table 3.1 has similar properties as DCKAP. Protocols in [5,93] do not provide fault tolerance property and may fail to protect the true calculation of the conference-key if a malicious attempt occurs. Protocols in [93, 98] do not provide forward secrecy. In these protocols, the distribution of the sub-key is realized by using the key pair that was distributed by the server. For this reason, if any of the key pair of a participant is compromised, the group keys that are generated by this key pair are also compromised. For dynamic group operations, [98] and [5] only provide operations for join and leave operations. Therefore, if merge and divide condition arise, the total effort for updating the conference-key is almost equal to the effort executing the protocol for the whole participant set. Since there is no arbitrary operation defined, the protocol in [93] does not provide efficient solution for leave and divide operations rather than re-executing the protocol. Detailed performance analysis of further operations is given in the subsequent sections.

3.2. DCKAP:A Dynamic Conference Key Agreement Protocol

In this section, we present DCKAP. DCKAP consists of two parts, ICKAP and ACKA operations. Let $\mathcal{U} = \{U_1, U_2, \ldots, U_n\}$ be the set of participants for a group key computation by using DCKAP. The activity diagram of DCKAP in UML notation is presented in Figure 3.1. As shown in the diagram, DCKAP consists of ICKAP and ACKA operations. In ICKAP, first, each participant generates and broadcasts temporary public key to other participants in the group. Next, each participant U_i verifies incoming temporary public keys by using their signature parameters. If any inconsistency is detected during verifications, owner of the temporary public key is reported as potential malicious participant. After temporary public keys are exchanged and verified, participants compute their own secret keys and broadcast these keys with signature parameters. If fault is detected during the verification of any secret key, owner of the secret key is marked as potential malicious participant. Potential malicious participants detected in either temporary public key verification or secret key verification are re-verified and if necessary extracted by using the fault detection

and correction function. This function is executed whenever the potential malicious attempt is detected. Finally, remaining honest participants execute the group key. Until the end of the communication session, the computed group key can be updated by using ACKA operations. ACKA operations provide four dynamic group operations, namely join, leave, merge and divide. With respect to the used ACKA operation, some of the participants are selected as active participants to update group key. Such active participants fully or partially execute ICKAP.

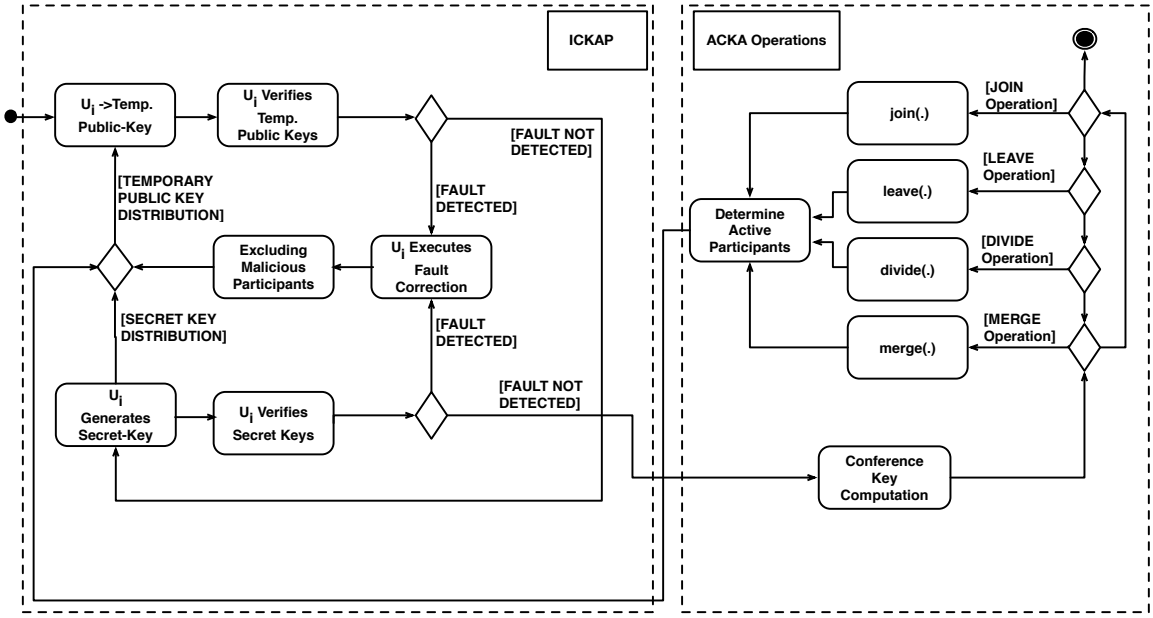

Figure 3.1. Activity Diagram of DCKAP in UML Notation

3.2.1. Initial Conference-Key Agreement Protocol

This section introduces the improved version of the protocol in [14]. In the proposed protocol, the sub-key generation proposed in Tseng's protocol is modified to handle ACKA operations and efficient fault correction.

Definition 3.1. *Temporary Public-Key Distribution Function, $TPKD(\cdot)$. In order to provide forward secrecy, each participant generates temporary public-key and their signatures as shown in Figure 3.2*

1 Each participant $U_i \in \mathcal{U}$ randomly selects a short-term secret keys t_i and $v_i \in Z_q^*$.

2 Then, U_i computes and broadcasts the following parameters:

$$T_i = g^{t_i} \ mod \ p, \ A_i = g^{v_i} \ mod \ p, \ B_i = v^{-1}(H(T_i, M) - A_i x_i) \ mod \ q$$

Figure 3.2. Temporary Public-Key Distribution Function, $TPKD(\cdot)$

Definition 3.2. *Secret Distribution and Commitment Function, $SDC(\cdot)$. Upon receiving all broadcast messages, (T_j, A_j, B_j) $(1 \leqslant j \leqslant n, \ j \neq i)$, from other participants in the group. Each participant verifies the incoming temporary public keys. Then, each participant U_i generates and broadcasts the secret key as shown in in Figure 3.3*

1 Each participant U_i checks that T_j is really issued by U_j by using the equation $g^{H(T,M)} = y_j^{A_j} A_j^{B_j} \ mod \ p$.

2 Then, U_i validates whether T_j is the generator of subgroup G_q, by using $2 \leqslant T_j \leqslant p-1$ and $T_j^q mod \ p$.

3 If these two checks hold, the sub-key is generated by using $K_i = (g^{t_i t_{i+1}} \ mod \ p)$ $mod \ q$. $R_i \in Z_q$, and $S_i \in Z_q^*$ are randomly selected.

4 Otherwise, U_i broadcasts $V_{k,j} = "failure"$ and executes $FDC(\cdot)$ function.

5 U_i constructs a polynomial $h_i(x)$ (over Z_q) with degree n that passes points $(j, (T_j^{R_i}$ $mod \ p) \ mod \ q)$, $(1 \leqslant j \leqslant n)$, and $(0, K_i)$ by U_i. Then, U_i computes and broadcasts the following:

$$\omega_{i,j} = h_i(n+j) \ mod \ q, \ 1 \leqslant j \leqslant n, \ \alpha_i = g^{R_i} \ mod \ p,$$

$$\gamma_i = g^{S_i} \ mod \ p, \ \delta_i = S_i^{-1}(H(K_i, M) - \gamma_i x_i) \ mod \ q.$$

Figure 3.3. Secret Distribution and Commitment Function, $SDC(\cdot)$

Definition 3.3. *Secret Key Computation and Verification Function, $SKCV(\cdot)$. Upon receiving $\omega_{j,l}$, $1 \leqslant l \leqslant n$, and α_j, each participant U_i computes and verifies the secret keys of other participants in the group as shown in Figure 3.4*

1. Each participant U_i uses his short-term secret key t_i to reconstruct the polynomial $h'_j(x)$ (over Z_q) with degree n that passes points $(n+l, \omega_{j,l})$, $1 \leqslant l \leqslant n$, and $(i, (\alpha_j^{t_i}$ $mod\ p)\ mod\ p)$.
2. Let $K'_j = h'(0)\ mod\ q$. Then, U_i checks whether $g^{H(K'_j,M)} = y_j^{\gamma_j}\gamma_j^{\delta_j}\ mod\ p$ holds or not. If the verifications are hold, U_i broadcasts $V_{k,j} =$ "*success*". Otherwise, U_i broadcasts $V_{k,j} = "failure"$ and executes $FDC(\cdot)$ function.

Figure 3.4. Secret Key Computation and Verification Function, $SKCV(\cdot)$

Definition 3.4. *Fault Detection and Correction Function, $FDC(\cdot)$. Broadcast messages of participants are verified in both $TPKD(\cdot)$ and $SKCV(\cdot)$ functions. According to these verifications, for each broadcast messages, the fault detection and correction is applied. Each participant pair U_i and U_j are marked as possible malicious participants if there exists a value $V_{i,j} = "failure"$ in the verification matrix. Broadcast messages of possible malicious participants are re-verified by honest participants. Steps of $FDC(\cdot)$ function are as shown in Figure 3.5*

1. Any participant U_k that broadcasts a message $V_{k,j} = "failure"$ cannot be the verifier in the second control of the broadcast message of U_j whether it is honest or not.
2. Any participant U_j with verification value $V_{k,j} = "failure"$, after the re-verification of broadcast messages by honest participant U_l, where $1 \leqslant l \leqslant n$ and $l \neq i \neq j$ the verification result is still $V_{l,j} =$ "$failure$", is removed from $\mathcal{U}$. Otherwise, U_k is removed from $\mathcal{U}$.
3. For each removed malicious participant U_j, U_{j-1} (if it is not malicious participant) randomly selects re-executes DCKAP from $TPKD(\cdot)$ function. According to the new broadcast messages, U_{j-2}, re-executes ICKAP from $SKCV(\cdot)$ function.

Figure 3.5. Fault Detection and Correction Function, $FDC(\cdot)$

Definition 3.5. *Conference Key Computation Function, $CKC(\cdot)$. If there is no fault detected, then all participants in the group $\mathcal{U}$ compute the conference key as shown in Figure 3.6*

1 Each participant $U_i \in \mathcal{U}$ computes the conference key as follows:

$$K' = (K_1 K_2 \cdots K_{m-1} K_m \ mod \ p) \ mod \ q \tag{3.1}$$

$$= (g^{t_1 t_2 + t_2 t_3 + \cdots + t_{m-1} t_m + t_m t_1} \ mod \ p) \ mod \ q \tag{3.2}$$

Figure 3.6. Conference Key Computation Function

When the conference key is updated, participants may re-execute ICKAP to generate new temporary-keys. Since the temporary private keys are selected randomly, such operations are called re-randomization. The formal definition of re-randomization is as follows:

Definition 3.6. *Re-randomization. Let $U_i \in \mathcal{U}$ be the participant that has already involved into the execution of ICKAP with temporary public - private key pair $(g^{t_i} \ mod \ p, t_i)$. The re-execution of ICKAP by U_i to obtain new key pair $(g^{t_i'} \ mod \ p, t_i')$, where $t_i \neq t_i'$, is called re-randomization.*

Tseng's protocol was designed for static groups. Therefore, any modification in the set of participants (i.e. fault correction) results with the re-execution of the protocol. If the set of participants changes frequently, then participants spend most of their time for protocol execution and the key computation performance of the protocol decreases. In DCKAP, Tseng's protocol has been modified. The short-term secret generation is realized by the equation $K_i = (g^{t_i t_{i+1}} \ mod \ p)$ instead of using random selection. The modified equation provides a relationship among consecutive participants. By using this assumption, it is possible to update the conference-key without re-executing the protocol. Therefore, in DCKAP, any single change causes at most two protocol re-executions, which provides better performance in the $FDC(\cdot)$ function. Further details are given in the performance analysis section.

In order to provide better performance in fault correction, this approach can be extended for dynamic group operations, which are defined as Auxiliary Conference-Key Agreement (ACKA) operations throughout the chapter.

3.2.2. Auxiliary Conference Key Agreement Operations

Dynamic group operations are used for updating the conference-key of a session without re-executing the protocol for each participant in the set of participants. In this chapter, these operations are called as ACKA operations, which provide the following functionalities:

(i) Participant join
(ii) Participant leave
(iii) Merge Conference Sessions
(iv) Divide a session into sub-conference sessions

For each possible modification in the set of participants, at least one participant re-randomizes the sub-secret by re-executing ICKAP. Details of ACKA operations are given in the following sub-sections.

Definition 3.7. *Join Function, join*$(\cdot)$*. This function is used to handle the mass and single join of new participants into the previously established conference session. Let* $m \geqslant 1$ *be the number of joining participants for the participant set* $\mathcal{U} = \{U_1, U_2, \ldots, U_n\}$*. Then, the resulting participant set is* $\mathcal{U} = \{U_1, U_2, \ldots, U_{n-1}, U_n, U_{n+1}, \ldots, U_{n+m-1}, U_{n+m}\}$*. Function for join operation is as shown in Figure 3.7.*

1 $U_i \in \{U_1, U_2, \ldots, U_{n-1}\}$ broadcasts the messages of $TPKD(\cdot)$ and $SDC(\cdot)$ functions of previous DCKAP execution. The participant U_{n-1} only broadcasts its message in $TPKD(\cdot)$ function, $(T_{n-1}, A_{n-1}, B_{n-1})$

2 $U_i \in \{U_n, U_{n+1}, \ldots, U_{n+m}\}$ executes ICKAP from the $TPKD(\cdot)$ function. Therefore, U_n re-randomizes its temporary-key and all participants $U_i \in \{U_{n+1}, \ldots, U_{n+m}\}$ generates their temporary public-keys such as $g^{t'_n}, g^{t_{n+1}}, g^{t_{n+2}}, \ldots, g^{t_{n+m}}$, respectively. Then, each participant $U_i \in \{U_{n+1}, \ldots, U_{n+m}\}$ broadcasts the new messages, (T_i, A_i, B_i) for $n \leqslant i \leqslant n+m$.

3 After the verification of the temporary public keys is completed, if no fault is detected, each $U_i \in \{U_{n-1}, U_n, U_{n+1}, \ldots, U_{n+m}\}$ executes $SDC(\cdot)$ function. The participant U_{n-1} has to execute this function, because, the participant U_n re-randomized its temporary key in the previous execution of $TKPD(\cdot)$ function. Therefore, the participants in $\{U_{n-1}, U_n, U_{n+1}, \ldots, U_{n+m}\}$ have the corresponding sub-secrets, $(g^{t_{n-1}t'_n}\ mod\ p)\ mod\ q$, $(g^{t'_n t_{n+1}}\ mod\ p)\ mod\ q$, $(g^{t_{n+1}t_{n+2}}\ mod\ p)\ mod\ q, \cdots, (g^{t_{n+m-1}t_{n+m}}\ mod\ p)\ mod\ q$. At the end of $SCD(\cdot)$ function, each participant $U_i \in \{U_{n-1}, \ldots, U_{n+m}\}$ broadcasts the messages $(\omega_{i,1}, \omega_{i,2}, \omega_{i,3}, \ldots, \omega_{i,n-1}, \omega_{i,n}, \omega_{i,n+1}, \ldots, \omega_{i,m+n-1}, \omega_{i,m+n}, \alpha_i, \gamma_i, \delta_i)$

4 If no fault is detected during the verification of second broadcast messages, each participant $U_i \in \{U_1, U_2, \ldots, U_{n+m}\}$ computes the conference-key as by using the $CKC(\cdot)$ function

Figure 3.7. Join Function

Definition 3.8. *Leave Function, leave*$(\cdot)$. *This function is used for re-randomizing the conference-key in case of a single or mass leave. By using this operation, it is possible to update the key without re-executing ICKAP for each participant in the session. If the set* $\mathcal{U}' = \{U_1, \ldots, U_m\}$, *where* $\mathcal{U}' \subseteq \mathcal{U}$ *and* $|\mathcal{U}'| \geqslant 1$, *is the set of leaving participants, then procedure for leave operation is as shown in Figure 3.8.*

```
for Each participant U_{j-1} ∈ U' do
    If U_j is the leaving participant, U_{j-1} ∈ U − U', re-executes the TPKD(·)
    function in ICKAP. Then, U_{j-1} ∈ U − U' generates a new temporary key as
    g^t_{j-1} mod p and broadcasts the key (T_i, A_i, B_i) by executing TPKD(·) function.
    Otherwise, if U_{j+1} is the leaving participant, U_{j-1} waits for the
    re-randomization in TPKD(·) function. If no fault occurs in the verification of
    new broadcast messages, then each U_j and U_{j-1} ∈ U − U' execute SCD(·)
    function.
    Each U_j and U_{j-1} ∈ U − U' generates the new sub-secret values according to
    the new temporary keys and broadcasts the messages (ω_{i,j}, α_i, γ_i, δ_i).
    Upon receiving the new broadcast messages, if no faults detected, all of the
    participant in the modified set, U − U', compute the conference-key by using
    new sub-key values as given in ICKAP.
end
```

Figure 3.8. Leave Function

Definition 3.9. *Merge Function, merge*$(\cdot)$*. Merge function is used for merging conference sessions to obtain a larger session. Assume that there exist k sessions, where* $k > 2$ *that participants of each session agreed on different conference keys by using ICKAP. These sessions are merged to form a bigger session by using the merge function. Let* $\mathcal{U} = \{\mathcal{U}_1, \mathcal{U}_2, \ldots, \mathcal{U}_k\}$ *be the set of participant sets to be merged, the function is operated as shown in Figure 3.9.*

```
foreach session 𝒰_j ∈ 𝒰 do
    foreach participant U_i ∈ 𝒰_j do
        If U_i is the last participant of any participant set 𝒰_j, where 1 ⩽ j ⩽ k, then
        this participant re-executes the TPKD(·) function of ICKAP and
        broadcasts the message (T'_i, A'_i, B'_i).
        Otherwise, each U_i just broadcasts the messages that they generated in their
        previous execution of TPKD(·) function.
        If no fault is detected for the broadcast messages, then in the execution in
        SCD(·) function of ICKAP, if the participant U_i ∈ 𝒰_t is located just before
        the last participant or it is located as the last participant in their participant
        lists, then these participants executes the SCD(·) function of ICKAP.
        Otherwise, each participant broadcasts the existing SCD(·) function
        messages of previous execution of ICKAP. In case of malicious attempt
        detection, the fault detection and correction procedure is applied for each
        new broadcast message.
    end
end
Finally, all honest participants compute the conference-key by executing the using
CKC(·) function.
```

Figure 3.9. Merge Function

Definition 3.10. *Divide Function, divide*$(\dot{})$. *In this function the set of participant of a session is divided into sub-sessions. Assume that* $\mathcal{U}_1, \mathcal{U}_2, \ldots, \mathcal{U}_k$ *is the sets of participants after division. The divide function is operates as shown in 3.10.*

1 Each participant $U_i \in \mathcal{U}_j$, where $1 \leq j \leq k$, controls the participant U_{i+1} and U_{i+2} in $\mathcal{U}$ before division.

2 If U_i, U_{i+1} and U_{i+2} are in the same session after division, then U_i does nothing.

3 If $U_i \in \mathcal{U}_j$ and $U_{i+1} \in \mathcal{U}_t$, where $t \neq j$, then U_i re-executes ICKAP from $TPKD(\cdot)$ function.

4 If $U_{i+2} \in \mathcal{U}_j$ and $U_i, U_{i+1} \in \mathcal{U}_t$, where $t \neq j$, then U_i re-executes ICKAP from $SCD(\cdot)$ function.

5 If any fault is detected for any sub-session during the execution of the function, then the $FDC(\cdot)$ function is executed for the subsession that the fault is detected.

6 Each honest participant computes the conference-key for the sub-session that the participant is attended.

Figure 3.10. Divide Function

3.3. Security Analysis of DCKAP

In this section, we give the security analysis of DCKAP. First, we show that ICKAP, together with ACKA operations provides the same security properties and the same resistance against known attacks defined in [14] and [6]. Moreover, we prove that ACKA operations are secure against the attack proposed in [18].

3.3.1. Correctness, Fault Tolerance and Forward Secrecy

The correctness property is used to show that if all of the participants follow the protocol, then each participant calculates the correct key as mentioned in [6]. DCKAP provides the correctness property for the participant set $\mathcal{U} = \{U_1, U_2, \ldots, U_n\}$ as follows.

Theorem 3.1. *Correctness. If all of the participants in $\mathcal{U}$ fully follows the protocol, then the key computed by ICKAP is common.*

Proof. Each $U_j \in \mathcal{U}$ can verify the broadcast messages (T_i, A_i, B_i, M) of U_i in $SCD(\cdot)$ function by using the verification equations $g^{H(T_j,M)} = y_j^{A_j} A_j^{B_j}$, $T_j^q mod p = 1$ and $2 \leqslant T_j \leqslant p$.

Since A_i, $H(T_j, M) \in Z_q$ is unique, a received temporary-key T_i must be the same for all participants. After the verification of the temporary-key, each U_j computes the sub-secret $K_j = (g^{t_j t_{j+1}} \ mod \ p) \ mod \ q$ and broadcasts the message $(\omega_{j,1}, \omega_{j,2}, \omega_{j,3}, \ldots, \omega_{j,n}, \alpha_j, \gamma_j, \delta_j)$. Then, each $U_k \in (U)$ computes and verifies the K_j for each participants as described in $SKCV(\cdot)$ function of ICKAP. Since for $H(K_i, M) \in Z_q$ is unique fixed γ_i and δ_i, all participants compute the same K_i. Thus, the conference-key computed by the participants is common. □

In the terms of ACKA operations, we assume that there exists a correct conference-key, which was generated by using ICKAP. The updated key as a result of ACKA operation is also correct if the following corollary holds.

Corollary 3.1. *Correctness for ACKA Operations. If all participants, including the new participants in join and merge operations are honest and the conference key was correctly calculated by ICKAP, then the correctness is preserved after ACKA operations.*

Proof. Assume $\mathcal{U} = \{U_1, U_2, \ldots, U_n\}$ is the set of participants. Then, participants that execute ICKAP perform the following operations:

(i) Join: Each new participant $U_i \in \{U_{n+1}, \ldots, U_{n+m}\}$ and U_n execute ICKAP from $TPKD(\cdot)$ function and the participant U_{n-1} executes ICKAP from $SDC(\cdot)$ function.

(ii) Leave: For each leaving participant U_i, participant U_{i-1} executes ICKAP from $TPKD(\cdot)$ function and participant U_{i-2} execute ICKAP from $SDC(\cdot)$ function.

(iii) Merge: Let $\mathcal{U} = \{\mathcal{U}_1, \mathcal{U}_2, \ldots, \mathcal{U}_k\}$ be the conference sessions to be merged, the last participants of each session executes ICKAP from the $TPKD(\cdot)$ function. The participants that are located just before the last participant of each session execute ICKAP from $SDC(\cdot)$ function.

(iv) Divide: For each participant $U_i \in \mathcal{U}$, if U_i is not in the same session with U_{i+1} after the division operation, then U_i re-executes ICKAP from from $TPKD(\cdot)$ function. Otherwise, if U_i and $U_{i+1} \in \mathcal{U}_k$ and $U_{i+2} \notin \mathcal{U}_k$ after the divide operation, then U_i re-executes ICKAP from $SDC(\cdot)$ function.

For all executions of $TPKD(\cdot)$ and $SDC(\cdot)$ functions, since for fixed A_i, $H(T_j,M) \in Z_q$ is unique and for fixed γ_i and δ_i, $H(K_i,M) \in Z_q$ is unique. Therefore, all of the participants compute the same updated key. Thus, the correctness property is preserved for ACKA operations. □

Another important property for the security analysis of DCKAP is the fault-tolerance. The fault-tolerance is used to detect and correct the malicious attempt during the execution of a conference-key agreement protocol and is defined in [6]. Tzeng also proposed a detailed analysis for the protocol, which is in the same study. For this analysis, we extend the same assumption for two broadcast messages.

Theorem 3.2. *Fault-Tolerance. A conference-key agreement protocol has fault-tolerance property if the following two conditions do not hold:*

(i) A malicious participant can cheat the honest participants by sending wrong key values.
(ii) A malicious participant can cheat the honest participants by identifying an honest participant as a possible malicious participant.

Proof. For the first case, assume that the participant U_i is a malicious participant. If U_i tries to cheat other participants while executing the $TPKD(\cdot)$ function of ICKAP, then the broadcast messages (T_i,A_i,B_i,M) is verified by other participants in the session with $g^{H(T_i,M)} = y_i^{A_i}A_j^{B_i} mod p$, $T_i^q mod\ p = 1$ and $2 \leqslant T_i \leqslant p$. If T_i value of U_i is wrong, at least one of the participant cannot verify T_i. If U_i tries to cheat in $SDC(\cdot)$ function of ICKAP, the broadcast message, $(\omega_{i,1},\omega_{i,2},\omega_{i,3},\ldots,\omega_{i,n},\alpha_i,\gamma_i,\delta_i)$, is verified by other participants with $g^{H(K_i',M)} = y_i^{\gamma_i}\gamma_i^{\delta_i}$ $mod\ p$. If the K_i value of U_i is wrong, then at least one of the honest participant can detect the mismatch during verification. If any malicious attempt of U_i is detected for either its first or second broadcast message, this participant is marked as possible malicious participant by using the verification matrix, $V_{j,i} =$ "$failure$" for all $1 \leqslant j \leqslant n$ and $i \neq j$. Thus, it is not possible to cheat participants by sending wrong key values in ICKAP.

If a participant U_i is marked as $V_{j,i} =$ "$failure$" by participant U_j and there exists a contrary value for some participant U_k as $V_{k,i} =$ "*success*", this participant's broadcast

messages are re-verified by other participant U_l. According to the re-verification, if U_i is malicious, then this participant will be excluded from the key computation. Otherwise, U_j will be excluded. Thus, it is not possible to identify an honest participant as malicious participant in ICKAP. Since these two conditions hold, ICKAP satisfies the fault-tolerance property. □

The proof above shows that fault-tolerance property is satisfied by using fault detection and correction function. The same condition holds for ACKA operations. There exists at most four type of broadcast messages according to the definitions of ACKA operations:

- Existing broadcast messages of a participant,
- New broadcast messages of an existing participant after re-randomization,
- New broadcast messages of a newly joining participant after its first protocol execution, and
- Existing broadcast message for $TPKD(\cdot)$ and new broadcast message for $SDC(\cdot)$ if the subsequent participant is re-randomizing.

Since fault detection and correction step executes for all cases listed above, ACKA operations provide fault-tolerance.

The last security property in this section is the forward secrecy. Forward secrecy is used to protect the conference-key against compromises of produced conference-keys, such as the compromise of participant's long-term key. To show that DCKAP provides forward secrecy, we will extend the Theorem 3.3 in [14] as follows.

Theorem 3.3. *Forward Secrecy. DCKAP provides forward secrecy under the difficulty of discrete logarithm problem.*

Proof. Assume that adversary A has the long-term key, x_i of any participant U_i and A can obtain K_i from either (γ_i, δ_i) or T_i and T_{i+1} by using the messages obtained in any of ICKAP or ACKA operations. For the first case, $\delta_i = S_i^{-1}(H(K_i, M) - \gamma_i x_i) mod q$. It can be easily

seen that δ_i has two unknown variables S_i and K_i. Therefore, it is computationally difficult to compute S_i by solving $\gamma_i = g^{S_i} mod p$ since this is the discrete logarithm problem. For the second case, solving either $T_i = g^{t_i} mod p$ or $T_{i+1} = g^{t_{i+1}} mod p$ to obtain $K_i = (g^{t_i t_{i+1}}\ mod\ p)$ $mod\ q$ is also discrete logarithm problem. So, A faces the difficulty of discrete logarithm problem again. Hence, DCKAP satisfies the forward secrecy property. □

3.3.2. Passive Attacks

In case of passive attacks, the attacker, who is not a member of the group, tries to obtain information from broadcast messages by eavesdropping the communication among participants without knowing the secrets x_i and K_i for any participant U_i. Tseng extended the proof in [6], which is based on adopting the Decisional Diffie-Hellman (DDH) assumption for eavesdropping attack. According to this assumption, the transcript of broadcast messages for any participant, U_i, are (T_i, A_i, B_i) and $(\omega_{i,1}, \omega_{i,2}, \omega_{i,3}, \ldots, \omega_{i,n}, \alpha_i, \gamma_i, \delta_i)$. We show that the real view $(T_i, A_i, B_i, \omega_{i,1}, \omega_{i,2}, \omega_{i,3}, \ldots, \omega_{i,n}, \alpha_i, \gamma_i, \delta_i)$ and the attacker's simulated view $(T_i', A_i', B_i', \omega_{i,1}', \omega_{i,2}', \omega_{i,3}', \ldots, \omega_{i,n}', \alpha_i', \gamma_i', \delta_i')$ on random variables $t_i' \in Z_q^*$, $v_i' \in Z_q^*$, $B_i' \in Z_q^*$, $\omega_{i,j}' \in Z_q (1 \leqslant j \leqslant n)$, $R_i' \in Z_q$, $S_i' \in Z_q^*$, $\delta_i' \in Z_q$ are computationally indistinguishable, where $T_i' = g^{t_i'}\ mod\ p$, $A_i' = g^{v_i'}\ mod\ p$ and $\gamma_i' = g^{S_i'}\ mod\ p$. The same transcripts and the corresponding assumptions hold for DCKAP but our protocol differentiates with the Tseng's in the generation of K_i. Instead of randomly selecting the value, we use $K_i = (g^{t_i t_{i+1}}\ mod\ p)$ $mod\ q$ = $(T_{i+1}^{t_i}\ mod\ p)\ mod\ q$. We know that the transcripts for both protocols are the same and x_i long-term private keys of any U_i is also secret. If we show that the K_i of any U_i is also secret, then we can make the same assumption in [14] for the security of DCKAP against passive attacks.

In [38], Boneh defined some examples of cyclic groups which DDH is shown to be intractable. One such group is given in Definition 2.1 (public parameters) as $p = 2q + 1$, q be large primes and g be the generator for the cyclic subgroup $G_q = \{i^2 | i \in Z_p^*\}$. According to this construction, the following two probability distributions are computationally indistinguishable:

- $(g^{t_i}, g^{t_{i+1}}, g^{t_i t_{i+1}})$, where t_i and t_{i+1} are randomly and independently chosen from Zq
- $(g^{t_i}, g^{t_{i+1}}, g^{c})$, where t_i, t_{i+1} and c are randomly and independently chosen from Zq

In other words, for any eavesdropper, E, the sub-secret, K_i, looks like a random variable in G_q. Hence, K_i is also secret for any E. Then, our problem, which is the computational indistinguishability of attackers view and the original view, is the same problem as in Tseng's protocol. See the security against passive attacks in [14] for further details.

3.3.3. Active Attacks

Another attack type for conference-key agreement protocols is the active attacks. The most known active attack is the impersonation attack or impersonator's adaptively chosen message attack. In this case, the attacker tries to impersonate the legal participant, U_i, by obtaining the temporary and long-term key from signatures. The random oracle model in [99] is adopted to prove the security of a signature schemes or key exchange protocols against this attack. The assumption of random oracle is that the one-way hash functions are accepted as a true random functions. In this analysis, we also concentrate on the broadcast messages (T_i, A_i, B_i) and $(\omega_{i,1}, \omega_{i,2}, \omega_{i,3}, \ldots, \omega_{i,n}, \alpha_i, \gamma_i, \delta_i)$ to show that they are existentially un-forgeable. The first part of this analysis is proved in *Theorem 1* in [14]. Since our protocol uses the same structure for $TPKD(\cdot)$ function of ICKAP, we prove that the improved version also provides the existentially un-forgeable property. Note that the K_i of U_i can also be calculated by U_{i+1}. However, the verification of $\omega_{i,j}$ values, for $1 \leqslant j \leqslant n$ can be realized by $g^{H(K'_j, M)} = y_j^{\gamma_j} \gamma_j^{\delta_j} \ mod \ p$. The equation together with the γ_i and δ_i parameter states that if any malicious adversary, MA, tries to impersonate the participant U_i, then MA has to obtain the long-term secret key, x_i. The following theorem shows that DCKAP is secure against the impersonation attack.

Theorem 3.4. *Any malicious adversary MA cannot compute the valid ω_i of any user U_i in the random oracle model since the discrete logarithm problem is intractable.*

Proof. Assume that α_i, γ_i and δ_i as a signature of $\omega_{i,j}$, where $1 \leqslant j \leqslant n$. Then the proof follows from the forking lemma in [100]. Under the random oracle model, $H(K_i, M)$ is an

independent random variable from K_i and M. Suppose that MA without knowing K_i or U_{i+1} with knowing K_i can impersonate U_i to sign $K_i = g^{t_i t_{i+1}}$ with a non-negligible probability ε. For any (ω_i, M), the MA can generate two valid signatures $(\alpha_i, \gamma_i, \delta_i)$ and $(\alpha_i, \gamma_i, \delta'_i)$, where $\alpha_i = g^{R_i}$ *mod* p, $\gamma_i = g^{S_i}$ *mod* p, $\delta_i = S_i^{-1}(H(K_i, M) - \gamma_i x_i)$ *mod* q, and $\delta'_i = S_i^{-1}(H'(K_i, M) - \gamma_i x_i)$ *mod* q. Therefore, it is computationally difficult to obtain x_i by solving equations δ_i and δ'_i under the random oracle model, since the discrete logarithm problem is computationally difficult. □

3.3.4. Key Freshness

The major vulnerability of dynamic groups is as shown in [18]. They proposed several attack scenarios for the protocol in [17]. According to these scenarios, if none of the sub-keys are refreshed after the modification of participant set, it is possible to retrieve former and subsequent conference-keys. Therefore, the backward confidentiality and forward confidentiality properties are violated. The formal definition of key freshness is as given below:

Definition 3.11. *If a conference-key agreement protocol achieves both backward confidentiality and forward confidentiality, then the key generated by using this protocol is called fresh.*

The following lemmas show that our protocol provides security against these attack scenarios.

Lemma 3.1. *Under the difficulty of computing the discrete logarithm problem, leave operation does not violate the forward confidentiality.*

Proof. Let $\mathcal{U} = \{U_1, U_2, U_3, \ldots, U_n\}$ be the set of participants before the divide operation. Assume that a leaving participant, U_i can obtain the update key, K' after leaving. Let $K = (g^{t_1 t_2 + \ldots + t_{i-1} t_i + t_i t_{i+1} + \ldots + t_n t_1}$ *mod* $p)$ *mod* q be the key before leave operation. Let $K' = (g^{t_1 t_2 + \ldots + t_{i-2} t'_{i-1} + t'_{i-1} t_{i+1} + \ldots + t_n t_1}$ *mod* $p)$ *mod* q be the updated key. The $g^{t_{i-2} t_{i-1} + t_{i-1} t_i + t_i t_{i+1}}$ part

of K is replaced $g^{t_{i-2}t'_{i-1}+t_{i-2}t'_{i-1}}$ in the leave operation. Therefore, it is computationally difficult to compute new key by using the old key K by solving the equation $K_{i-2} = (g^{t_{i-2}t'_{i-1}}$ $mod\ p)\ mod\ q$ or $K_{i-1} = (g^{t'_{i-1}t_{i+1}}\ \ mod\ p)\ mod\ q$ since this is the discrete logarithm problem. □

Lemma 3.2. *Under the difficulty of computing the discrete logarithm problem, divide operation does not violate the forward confidentiality.*

Proof. Let $\mathcal{U} = \{U_1, U_2, U_3, \ldots, U_n\}$ be the set of participants before the divide operation. Let $K = (g^{t_1t_2+\ldots+t_{k-1}t_k+t_kt_{k+1}+\ldots+t_nt_1}\ mod\ p)\ mod\ q$. Assume that the set divided into two subsets as $\mathcal{U}_1 = \{U_1, U_2, \ldots U_k\}$ and $\mathcal{U}_2 = \{U_{k+1}, U_{k+2}, \ldots U_k\}$. Let $U_i \in \mathcal{U}_1$ try to obtain the updated key of $\mathcal{U}_2$. U_n re-randomizes the temporary key t_n as t'_n, then the updated key of $\mathcal{U}_2$ will be $K_2 = (g^{t_kt_{k+1}+\ldots+t_{n-1}t'_n+t'_nt_k}\ \ mod\ p)\ mod\ q$. Therefore, it is computationally difficult to obtain the conference key of $\mathcal{U}_2$ without knowing t'_n, since this is the discrete logarithm problem. □

Lemma 3.3. *Under the difficulty of discrete logarithm problem, join operation does not violate the backward confidentiality.*

Proof. Let $\mathcal{U} = \{U_1, U_2, U_3, \ldots, U_n\}$ be the set of participants before the join operation and U_{n+1} be the joining participant. Let $K = (g^{t_1t_2+\ldots+t_{n-1}t_n+t_nt_1}\ mod\ p)\ mod\ q$ and let $K' = (g^{t_1t_2+\ldots+t_{n-1}t'_n+t'_nt_{n+1}+t_{n+1}t_1}\ \ mod\ p)\ mod\ q$ be the updated key before and after join operation, respectively. The $g^{+t_{n-1}t_n+t_nt_1}$ is the updated part of K. So, the joining participant U_i can obtain the old key by using the new key K' if participant solves the equation $K_n = (g^{t_nt_1}$ $mod\ p)\ mod\ q$ or $K_{n-1} = (g^{t_{n-1}t_n}\ \ mod\ p)\ mod\ q$ to obtain t_n. Therefore, it is computationally difficult to compute t_n by solving K_n and K_{n-1} is computationally difficult since this is the discrete logarithm problem. □

Lemma 3.4. *Under the difficulty of computing the discrete logarithm problem, merge operation does not violate the forward confidentiality.*

Proof. Assume that $\mathcal{U}_1 = \{U_1, U_2, \ldots U_k\}$ and $\mathcal{U}_2 = \{U_{k+1}, U_{k+2}, \ldots U_n\}$ be the two conference sessions to be merged.Let

$$K_1 = (g^{t_1 t_2 + \ldots + t_{k-1} t_k + t_k t_1} \ mod \ p) \ mod \ q$$

and

$$K_2 = (g^{t_{k+1} t_{k+2} + \ldots + t_{n-1} t_n + t_n t_{k+1}} \ mod \ p) \ mod \ q$$

be the corresponding conference-keys, respectively. Also, assume that $\mathcal{U} = \{U_1, U_2, U_3, \ldots, U_n\}$ be the set of participants after the merge operation. Then,

$$K = (g^{t_1 t_2 + \ldots + t_{k-1} t'_k + t'_k t_{k+1} + \ldots + t_{n-1} t_n + t'_n t_1} \ mod \ p) \ mod \ q$$

be the key after merge operation. $U_i \in \mathcal{U}_1$ tries to obtain the key K_2 of $\mathcal{U}_2$ by using K but the temporary key t_n is re-randomized as t'_n during the merge operation. Therefore it is computationally difficult to compute the K_2 without knowing the t'_n since this is the discrete logarithm problem. □

These four lemmas show that if there exists any modification in the set of participants, then the conference-key is updated by re-randomizing at least one of the secret short-term keys. Therefore, the forward confidentiality and backward confidentiality properties are preserved after these operations. Thus, the following theorem holds.

Theorem 3.5. *Under the difficulty of computing discrete logarithm problem, a conference-key is updated by using ACKA operations is always fresh.*

Proof. Lemmas 4.1 and 4.2 show that ACKA operations provide the forward confidentiality and Lemmas 4.3 and 4.4 proved that the forward confidentiality is also satisfied. Since both the forward and backward confidentialities are satisfied under the difficulty of discrete logarithm problem, ACKA operations always produce fresh keys

On the other way, let $\mathcal{U} = \{U_1, U_2, U_3, \ldots, U_n\}$ be the set of participant and $K = (g^{t_1 t_2 + \ldots + t_{n-1} t_n + t_n t_1} \ mod \ p) \ mod \ q$ be the corresponding key. Suppose that ACKA operations do not provide the key freshness. Also suppose that the participant U_i leaves the session for a while and then joins the session later. Since the resulting set of participant is the same, the resulting key must be the same. However, after the leave operation, the conference-key will be $K' = (g^{t_1 t_2 + \ldots + t_{n-2} t'_{n-1} + t'_{n-1} t_1} \ mod \ p) \ mod \ q$ for the set $\mathcal{U}' = \{U_1, U_2, U_3, \ldots, U_{n-1}\}$. When the U_n joins the session again, for $\mathcal{U}'' = \{U_1, U_2, U_3, \ldots, U_n\}$, the key will be $K'' = (g^{t_1 t_2 + \ldots + t_{n-2} t''_{n-1} + t''_{n-1} t_n + t_n t_1} \ mod \ p) \ mod \ q$. After these operations, the $\mathcal{U}'' = \mathcal{U}$ but $K'' \neq K$. Hence the contradiction. □

As a result, DCKAP provides the same security with the Tzeng's protocol and Tseng's protocol. Since, ACKA operations provides backward confidentiality and forward confidentiality, DCKAP provides key freshness.

3.4. Performance Analysis of DCKAP

In this section, we give the performance analysis of DCKAP. The performance analysis of DCKAP is based on the performance analysis of ICKAP and ACKA operations. During the analysis of DCKAP, we use the communications cost analysis and computational cost complexity for the performance criteria as defined in Section 2.2.

In addition to the general performance analysis criteria above, we use the number of protocol executions for ACKA operations and $FDC(\cdot)$ function of ICKAP. The number of protocol executions is the total number of participants that executes ICKAP while updating the conference-key.

The performance comparison of ICKAP with the Tseng's protocol is as follows. Both protocol have constant communications cost. There exist two rounds for communication in ICKAP and Tseng's protocol. In case of computation cost, ICKAP differs only in the $SDC(\cdot)$ function with the Tseng's protocol. The computation cost of Tseng's protocol is $\mathcal{C}_{comp} = (5n-2)T_{EXP} = O(n)T_{EXP}$. In ICKAP, only the time for modular exponential operations is

increased by one, $\mathcal{C}_{comp} = (5n-1)T_{EXP} = O(n)T_{EXP}$, which is a minor difference.

Finally, since there exists no difference for the parameters of broadcast messages, the message length is the same as Tseng's protocol:

$$\mathcal{C}_{length} = (n+2)|q| + 4|p| = O(m^2)|q| + O(m)|p|$$

Table 3.2. Comparison for Total Fault Correction Costs

Protocols	$\mathcal{C}_{comp}$	$\mathcal{C}_{length}$	$\mathcal{C}_{round}$
Tzeng's Protocol	$O(m^2)T_{EXP}$	$O(m^2)\|q\| + O(m)\|p\|$	$O(m)$
Tseng's Protocol	$O(m^2)T_{EXP}$	$O(m^2)\|q\| + O(m)\|p\|$	$O(m)$
DCKAP	$O(m)T_{EXP}$	$O(m)\|q\| + O(1)\|p\|$	$O(1)$

Since DCKAP is designed for the dynamic groups, it provides better performance while updating the conference-key from fault-tolerant point of view. Tseng's protocol only works on static groups. Hence, if any modification occurs in the participant set, then each participant in the updated set of participants executes ICKAP. Set modification can occur in either to exclude malicious participants or in ACKA operations. The comparisons for fault-correction is given in Table 4.3, where m is the number of participants in the set after the malicious participants are excluded.

The fault correction operations are used by ICKAP to exclude the malicious participant from the set of participants. Assume that there exists k malicious participants in $\mathcal{U}$ and $m-k \geqslant 2$. For each malicious participant U_i, the honest participant U_{i-1} re-executes ICKAP from $TPKD(\cdot)$ function and the participant U_{i-2} re-executes ICKAP from $SDC(\cdot)$ function. Then, for each malicious participant, there exist one execution of ICKAP from $TKPD(\cdot)$ and one execution from $SDC(\cdot)$ for re-randomization of the conference key. Therefore, the protocol execution cost for fault correction of ICKAP is $O(1)$ for each malicious participants. Since Tseng's protocol does not provide an efficient fault-correction mechanism, all of the participants has to re-execute the protocol to update the key. Since it is designed for static

groups, the protocol execution cost for fault correction of Tseng's protocol is $O(m)$, where m is the number of participants in the session.

The performance analysis of DCKAP based on ACKA operations is as follows. Assume that m is the number of participants in the session $\mathcal{U}$. The performance analysis for ACKA operations are given below:

- Join Operation: Assume that $k \geqslant 1$ is the number of new participants. During the participant join procedure, $k+1$ participants execute ICKAP from $TPKD(\cdot)$ function and the participant, which is located before the last participant, U_{m-1}, executes ICKAP from $SDC(\cdot)$ function. Since there exists $k+1$ full executions and the join operation has constant execution complexity, there exist $(k+1) \cdot O(1) = O(k)$ for join operation. Since the computation cost of ICKAP is $O(m)T_{EXP}$, the complexity of join operation is $\mathcal{C}_{comp} = O(km)T_{EXP}$.
- Leave Operation: Assume that $k \geqslant 1$ is the number of leaving participants for $m-k \geqslant 2$. Then, if all leaving participants are ordered consecutively as $U_{i+1}, U_{i+2}, \ldots, U_{i+k}$, then only U_i re-executes ICKAP from $TPKD(\cdot)$ function and U_{i-1} executes ICKAP from $SDC(\cdot)$ function, which is the best case for leave operation with $O(1)$ execution cost and computational complexity cost is $\mathcal{C}_{comp} = O(m)T_{EXP}$. Otherwise, for each leaving participant U_{i+1}, U_i re-executes the protocol from $TPKD(\cdot)$ function and U_{i-1} executes the protocol from $SDC(\cdot)$ function. There exists at most k re-executions of ICKAP. Then, the execution cost is $O(k)$. Regarding the modular exponentiations of ICKAP, the computational complexity cost of leave operation is $\mathcal{C}_{comp} = O(km)T_{EXP}$.
- Merge Operation: Assume that there exists k sessions and each session has m participants. According to the definition of merge operation, only the last participant for each session re-executes ICKAP and also participants before the last participants execute ICKAP from $SDC(\cdot)$ function, which means the merge operation also have constant protocol execution complexity. Therefore, the protocol execution cost of merge operation is $k \cdot O(1) = O(k)$ and overall complexity of merge operation is $\mathcal{C}_{comp} = O(k^2m)T_{EXP}$.

- Divide Operation: Let k be the number of sub-sessions after division and each session has m participants. If the participants of the same sub-session are ordered consecutively, then there exists m re-executions of ICKAP and m executions of ICKAP from $SDC(\cdot)$ function, which is the best case complexity with $O(k)$. Otherwise, for each participant pair U_i and U_{i+1}, if they belong to different sessions, U_i re-executes ICKAP and U_{i-k} executes ICKAP from $SDC(\cdot)$ function. This case is the worst case complexity for protocol executions with $O(k \cdot m)$. The complexity of divide operation, including the computational cost of ICKAP, is $\mathcal{C}_{comp} = O(km)$ and $\mathcal{C}_{comp} = O(k^2 m)$ for the best case and the worst case, respectively.

Table 3.3. Comparison of Dynamic Group Operations of Protocols for the Number of Protocol Executions

Operations	$\mathcal{C}_{comp}$				**Number of Participants or Sessions**
	Protocol 1 [93]	**Protocol 2 [98]**	**Protocol 3 [5]**	**DCKAP**	
Join	$O(k)$	$O(k)$	$O(k \cdot (m+k))$	$O(k)$	for k joining participants
Leave	NA	$O(k)$	$O(k \cdot (m-k))$	$O(k)$	for k leaving participants
Merge	$O(k)$	NA	NA	$O(k)$	for k sessions
Divide	NA	NA	NA	$O(k)$,	for k sessions

In Table 3.3, we present the comparison of protocols [5, 93, 98] and ACKA according to protocol execution cost for dynamic group operations. For convenience, protocols in [5, 93, 98] are named as Protocol 1, Protocol 2 and Protocol 3, respectively. As seen in Table 3.3, only DCKAP provide all of the dynamic group operations. Protocol 2 and Protocol 3 do not provide divide and merge operations. It is also given that Protocol 1 does not have an efficient operation for leave and divide rather than re-executing the protocol. The detailed analysis for comparisons is as follows:

- Assume that $\mathcal{U}_1 = \{U_1, U_2, \ldots U_k\}$ and $\mathcal{U}_2 = \{U_{k+1}, U_{k+2}, \ldots U_n\}$, where $\mathcal{U}_1 \cap \mathcal{U}_2 = \emptyset$, are the two conference sessions to be merged and $K_1 = (g^{t_1 t_2 + \ldots + t_{k-1} t_k + t_k t_1} mod p) mod q$ and $K_2 = (g^{t_{k+1} t_{k+2} + \ldots + t_{n-1} t_n + t_n t_{k+1}} mod p) mod q$ be the corresponding conference-keys, respectively. Since Protocol 2 and Protocol 3 do not provide merge operation, they simulate these operations either by using leave and join operations or they execute their initial key agreement protocols for $\mathcal{U}_1 \cup \mathcal{U}_2$. In both case, the computation cost of key update is larger than Protocol 1 and DCKAP.
- In case of the divide operation, the worst case complexity of protocol execution cost of ACKA is $O(k \cdot m)$, where k is the number of sessions and m is the number participants for each session. In addition, the best case cost for DCKAP is $O(k)$. Protocol 1, Protocol 2 and Protocol 3 do not provide arbitrary divide operation, therefore they are not comparable to DCKAP. Assume that $\mathcal{U} = \{U_1, U_2, \ldots U_n\}$ is the set to be divided and $K = (g^{t_1 t_2 + t_2 t_3 + \ldots + t_{n-1} t_n + t_n t_{k+1}} \ mod \ p) \ mod \ q$ is the corresponding conference-key. If the session is divided into two sub-sessions as $\mathcal{U}_1 = \{U_1, U_2, \ldots U_k\}$ and $\mathcal{U}_2 = \{U_{k+1}, U_{k+2}, \ldots U_n\}$, then Protocol 2 and Protocol 3 have to execute their initial key agreement protocols since they do not provide divide operations. For Protocol 1, it is possible to divide $\mathcal{U}$ into $\mathcal{U}_1$ and $\mathcal{U}_2$ by reverse execution of the protocol. However Protocol 1 is restricted in dividing the set of participants in arbitrary manner. Therefore, it provides only re-execution for leave and divide operations. Moreover, the number of communication rounds is $\mathcal{C}_{round} = logn$ in Protocol 1, which is $\mathcal{C}_{round} = 1$ in DCKAP.

Performance comparisons show that DCKAP has better performance in terms of fault correction. Moreover, DCKAP provides linear time computational complexity for ACKA operations.

3.5. Scalability Analysis of DCKAP

In this section, we compare the scalability analysis of DCKAP with other protocols [17, 101, 102] regarding mass join and mass leave. We first compared the scalability of protocols for fixed join and leave rate. Then, we analyzed the scalability of protocols for

mass join and mass leave of $50, 100, 150, 200$ and 250 participants. Simulations were carried out by using Python 3 and Charm Framework [103] that run on a MacBook Air 2012 early release with 250 GB SSD disk, 1.8GHz Intel Core i5 processor and 4GB 1600 MHz DDR3 RAM.

Table 3.4. Scalability Comparison of Protocols for 10% Join and Leave $(\times 10^6 ms)$

Protocols	**Operations**	**# of Participants**			
		500	**1000**	**2000**	**4000**
DCKAP in [16]	Join	2.7	10.6	42.1	167
	Leave	0.07	0.14	0.27	0.54
Tseng in [17]	Join	2.1	8.2	32	129
	Leave	0.03	0.05	0.11	0.21
AD-ASGKA in [101]	Join	0.59	2.2	8.5	33
	Leave	0.46	1.7	6.7	26
Zhang *et al.* in [102]	Join	2.4	9.3	36.7	146
	Leave	1.9	7.7	30.1	119

In Table 3.4, the scalability of protocols according to the fixed mass join and mass leave rates are given. The join rate is fixed for 10% for groups of $500, 1000, 2000$ and 4000 participants. According to the simulation results, the best performance is provided by the AD-ASGKA protocol in [101] for only the mass join operations. For mass leave operation, Tseng's protocol provides the best results. On the other hand, the worst performance for mass join belongs to DCKAP. Moreover, simulation results show that when the number of participants is doubled, the total communications and computational costs of each DGKAP are quadrupled for mass join. In terms of mass leave, protocol proposed by Zhang *et al.* has the worst performance. For mass leave, except the protocol in [102], the group size and the total communications and computational costs increase at the same rate. As a result, for the fixed mass join rates, asymmetric group key agreement protocols provide better performance than symmetric ones. However, in terms of fixed mass leave rates, symmetric group key agreement protocols provide better performance.

After the scalability analysis for fixed join rate, we analyze the scalability of protocols according to the 50, 100, 150, 200 and 250 joining and leaving participants as shown in Figures 3.11-3.18. In Figures 3.11-3.14, the scalability of mass join is presented. According to the Figures 3.11, 3.12 and 3.13, AD-ASGKA protocol provides the best performance. On the other hand, DCKAP has the worst performance for mass join. Therefore, computational cost overhead of arithmetic operations in DCKAP affects the total computational cost of the protocol. According to Figure 3.14, it is shown that if mass join rate is around 2.5% then, AD-ASGKA protocol has the worst performance. Moreover, it is also shown that the total communications and computational costs of protocols increase linearly with respect to the fixed mass joins and linearly increasing group size for protocols [17, 102] and DCKAP.

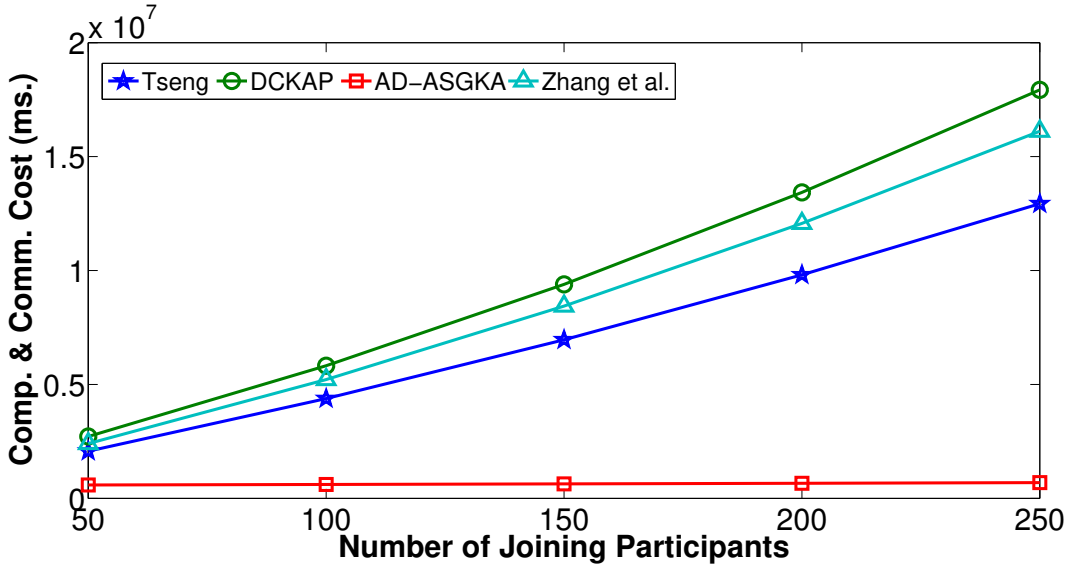

Figure 3.11. Scalability Comparison of Protocols for 50, 100, 150, 200 and 250 Joining Participants for Group of 500 Participants

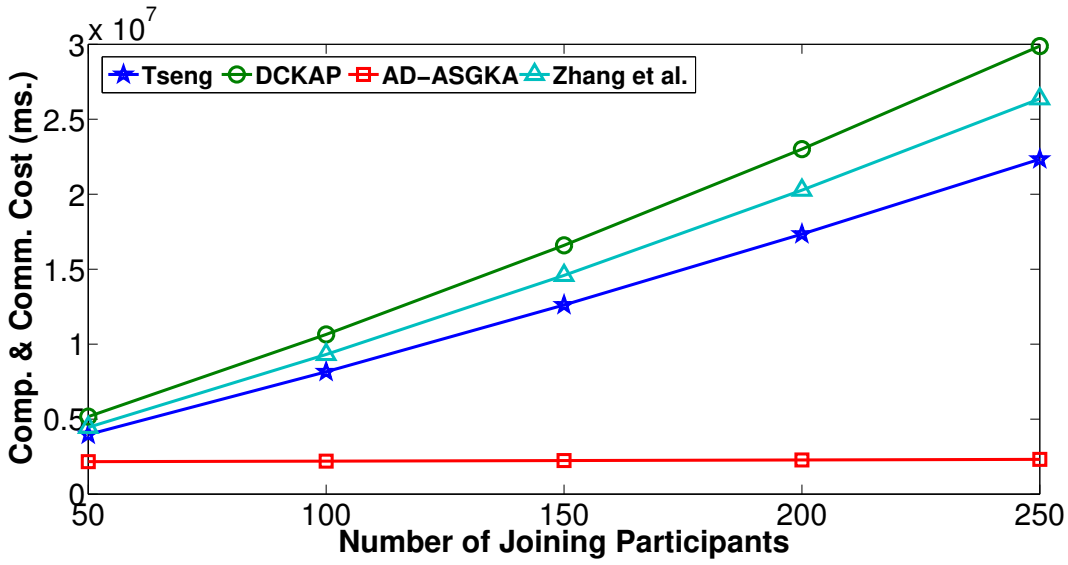

Figure 3.12. Scalability Comparison of Protocols for 50, 100, 150, 200 and 250 Joining Participants for Group of 1000 Participants

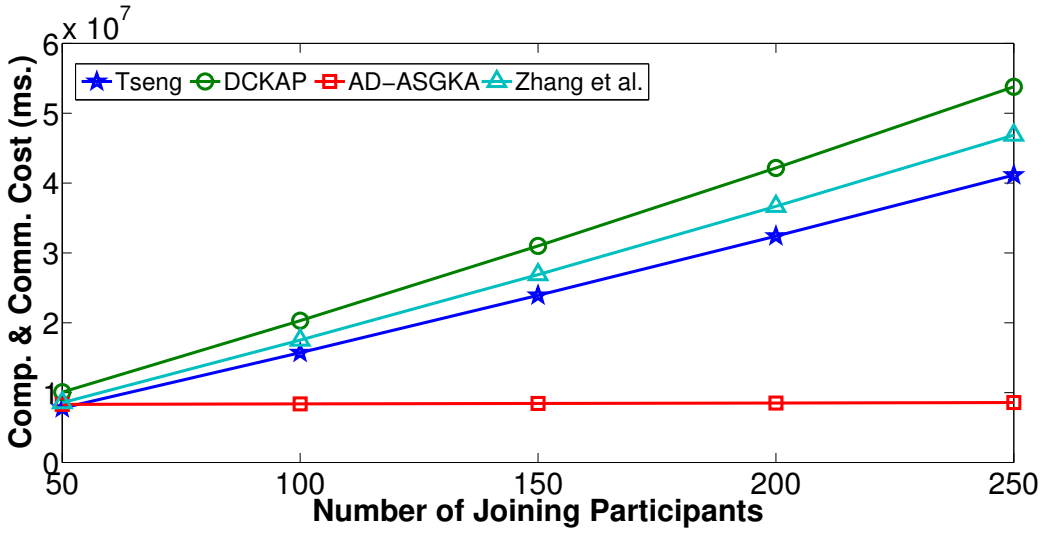

Figure 3.13. Scalability Comparison of Protocols for 50, 100, 150, 200 and 250 Joining Participants for Group of 2000 Participants

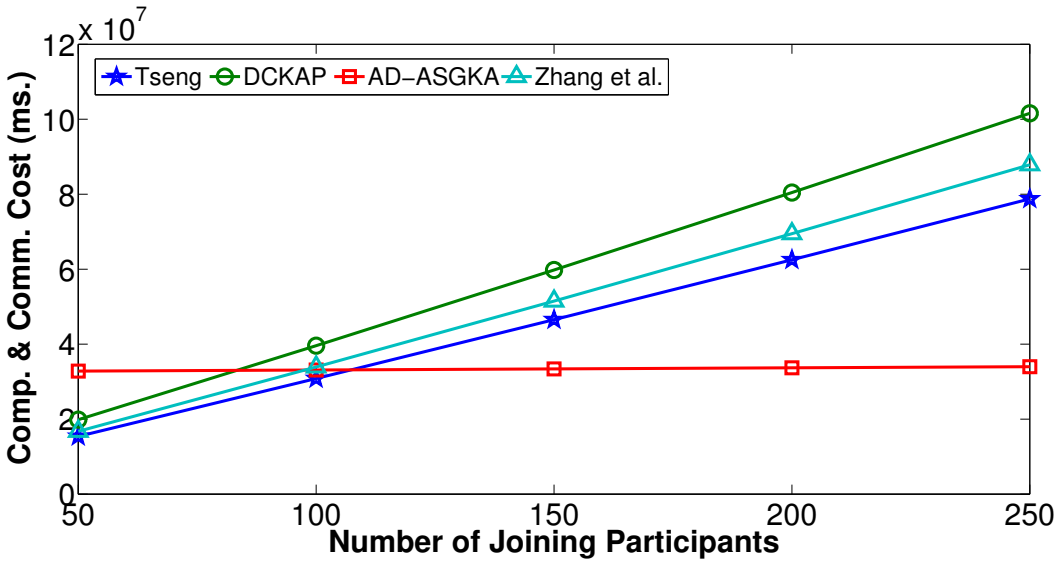

Figure 3.14. Scalability Comparison of Protocols for $50, 100, 150, 200$ and 250 Joining Participants for Group of 4000 Participants

Scalability analysis for mass leave operation of the protocols are presented in Figures 3.15-3.18. Simulation results show that Tseng's protocol has the best performance for mass leave operations. Moreover, symmetric protocols provide better performance than asymmetric protocols since they have constant computation costs in terms of signature generation (See performance analysis for details). Furthermore, Zhang *et al.* has the worst performance for mass leave.

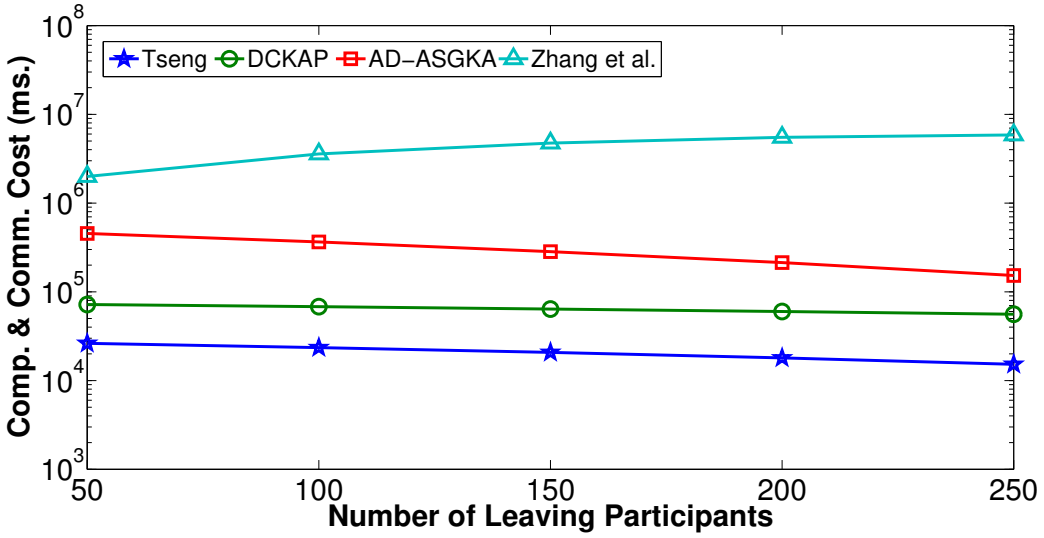

Figure 3.15. Scalability Comparison of Protocols for 50, 100, 150, 200 and 250 Leaving Participants for Group of 500 Participants

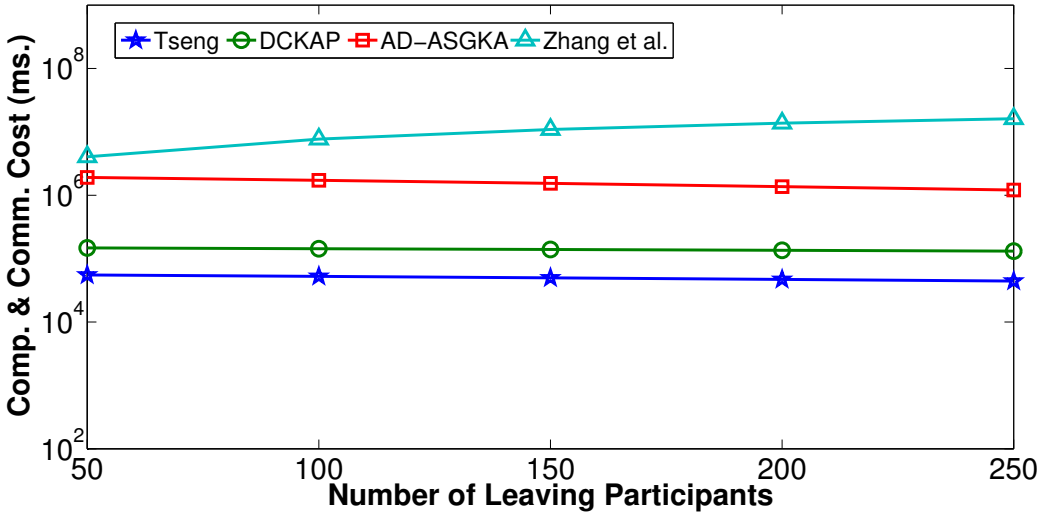

Figure 3.16. Scalability Comparison of Protocols for 50, 100, 150, 200 and 250 Leaving Participants for Group of 1000 Participants

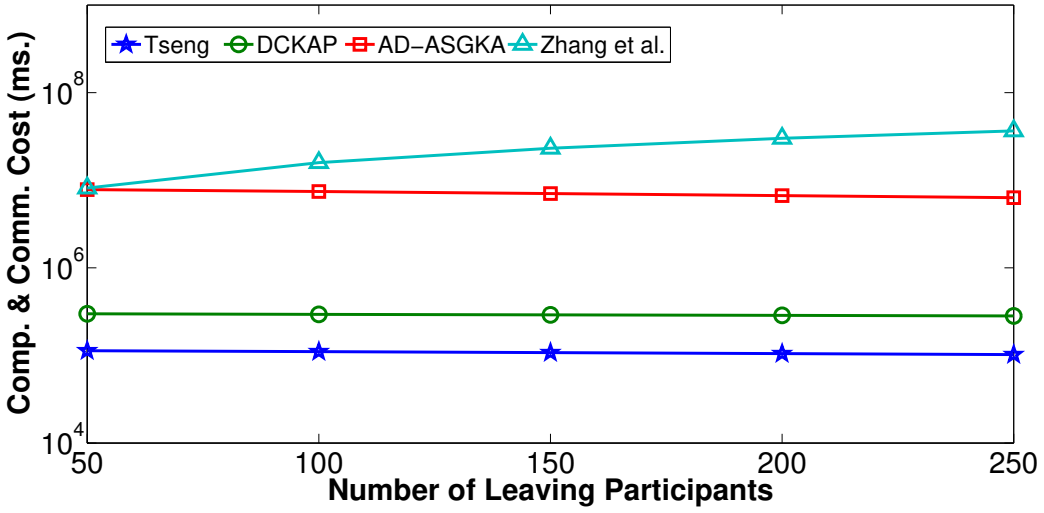

Figure 3.17. Scalability Comparison of Protocols for 50, 100, 150, 200 and 250 Leaving Participants for Group of 2000 Participants

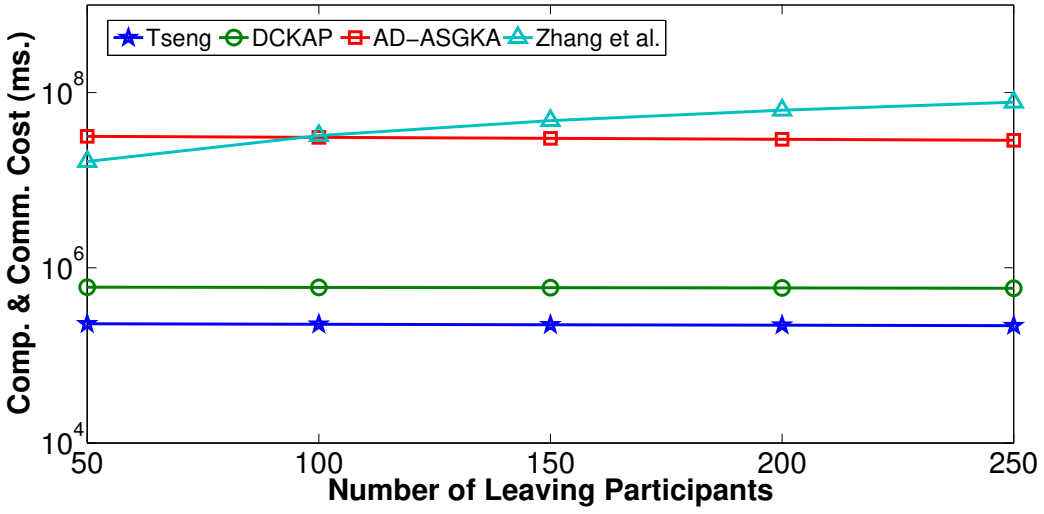

Figure 3.18. Scalability Comparison of Protocols for 50, 100, 150, 200 and 250 Leaving Participants for Group of 4000 Participants

As a consequence, we have compared the scalability of DCKAP via simulations. Simulation results show that average performance of DCKAP for mass join operations is the worst. On the other hand, for mass leave operations, DCKAP and Tseng's protocol provides better performance than asymmetric protocols. As defined in Section 3.3, Although Tseng's protocol has provided better results than DCKAP, as we mentioned in Section 3.3, Tseng's

protocol fails to provide backward confidentiality and forward confidentiality. Therefore, in addition to conference communication applications, DCKAP can be a good candidate for applications in which computation cost effective mass leaves are primary concern such as the efficient revocation of participant permissions in file sharing systems. As a result of DCKAP and simulations that we have presented in scalability analysis, we provide a new file sharing system called TT-SFSS in the following section.

3.6. TT-SFSS: A Three-Tier Secure File Sharing System with Efficient Participant Revocation, An Application of DCKAP

In this section, we propose a Three-Tier Secure File Sharing System, called TT-SFSS for providing an efficient approach for participant revocation in file sharing systems. The "Three Tiers" of TT-SFSS corresponds to the following encryption operations:

(i) Providing confidentiality by randomly generated File Confidentiality Keys (FCKs)
(ii) Providing attribute-based access control for for file confidentiality keys
(iii) Storing files in a directory called Keybag, which is encrypted via using DCKAP.

In contrast to the previously proposed secure file sharing systems, by using three-tier approach, TT-SFSS provides a novel two-level participant revocation mechanism, called Primary Revocation and Secondary Revocation. Primary revocation of TT-SFSS is realized by using DCKAP as a revocation mechanism. When a participant leaves file sharing, $leave(\cdot)$ of DCKAP is executed to update the group key. Secondary Revocation of TT-SFSS is the same as the previous secure file sharing systems [70, 79, 90]. When a participant leaves file sharing, then access permissions are revoked by using proxy re-encryption [81] and lazy re-encryption [89]. Details of the services and corresponding security and performance analysis are given in the rest of the section.

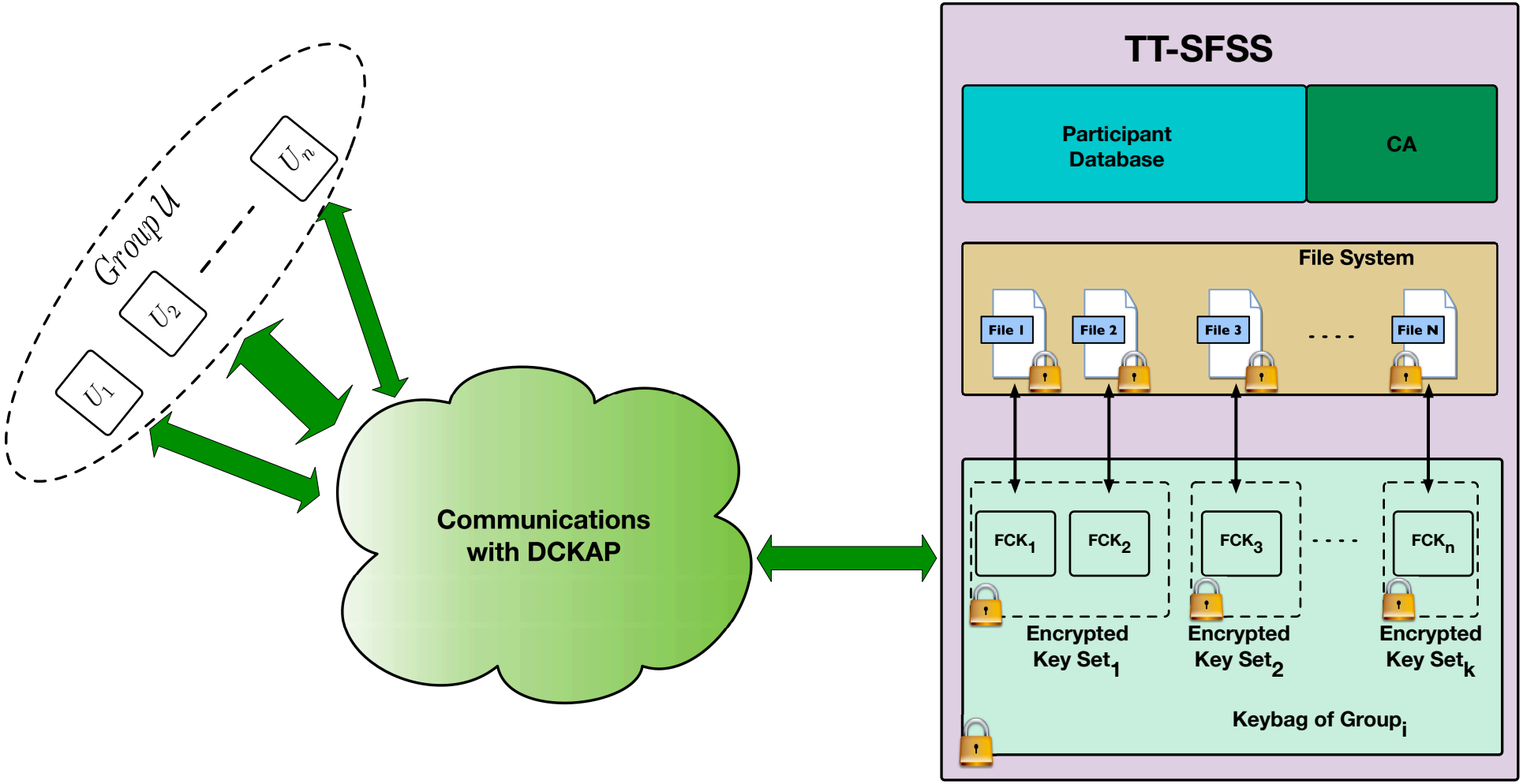

Figure 3.19. An Overall View of TT-SFSS

3.6.1. An Overall View of TT-SFSS

TT-SFSS components, which are shown in the overall structure of TT-SFSS in Figure 3.19, are as follows:

- Participant Groups: Let $\mathcal{U}_1, \mathcal{U}_2, \mathcal{U}_3, \ldots \mathcal{U}_N$ be the groups of TT-SFSS. Each group $\mathcal{U}_i$, where $1 \leq i \leq N$ has its own set of participants, $\mathcal{U}_i = \{U_1, U_2, U_3, \ldots, U_{|\mathcal{U}_i|}\}$.
- Certificate Authority (CA): After the registration of a new user is completed, each participant U_i has to generate and certify long-term key pairs in order to join file sharing. Then, the public key of each participant U_i is certified by using CA in PFSS.
- Communications with DCKAP: Participants in each participant group $\mathcal{U}_i$ computes group key by using DCKAP. The group key is used for encrypting the shared file while uploading to or downloading from TT-SFSS servers.
- Participant Database: TT-SFSS provides registration system for new users. After the registration is completed, users become the possible participants of file sharing groups in TT-SFSS. Login credentials and information of groups that a participant involved are stored in participant database in TT-SFSS.
- File System Files and File Confidentiality Keys: Proposed system has a file system in its cloud servers to organize the files as shared files. Shared files of a group can be expressed as $\mathcal{F} = \{F_1, F_2, F_3, \ldots, F_N\}$. A shared file $F_i \in \mathcal{F}$ could as well be a directory identifying a tree of shared files. Owner of the file $U_i \in \mathcal{U}$ encrypts the file to be shared by using randomly generated key, called file confidentiality key. Then, file confidentiality key is encrypted by using CP-ABE. Therefore, only the participants, who has the correct attirbutes can obtain the file confidentiality key. Moreover, shared files are stored in encrypted format in the TT-SFSS file system.
- Keybag: After the file to be shared is uploaded to the TT-SFSS server, file confidentialitiy keys are also uploaded and stored in a directory called Keybag. Keybag is used for storing all of the encrypted file confidentiality keys by using CP-ABE. Moreover, Keybag is an encrypted directory with AES-256 by using group key.

Let $\mathcal{U} = \{U_1, U_2, U_3, \ldots U_N\}$ be the set of registered participants, TT-SFSS operates as follows:

(i) File sharing in TT-SFSS starts with the execution of DCKAP for participants in $\mathcal{U}$ and a TT-SFSS instance $U_{TT-SFSS}$ to compute group key K.

(ii) Then, any participant U_i can share a file with other participants in $\mathcal{U}$. For instance, assume U_i wants to share a file F with participants in $\mathcal{U}$.

(iii) After the execution of DCKAP, group key K is computed. First, U_i encrypts F by using AES-256 with randomly generated File Confidentiality Key (FCK).

(iv) U_i initializes the access permission for FCK by encrypting with Ciphertext-Policy Attribute Based Encryption (CP-ABE) in [1] and stores encrypted FCK in a Keybag, which is a directory to store encrypted FCKs of a group.

(v) Keybag is encrypted with AES-256 by using the group key K and all transactions between Cloud Server of TT-SFSS and participants are realized by using K.

3.6.2. TT-SFSS Services

Conventional approach for participant revocation is a burden for file sharing systems due to the extra computational costs for re-encryption of shared files and distribution of new file confidentiality keys. One approach is to postpone the re-encryption process until the next update of a shared file. The postponement of file re-encryption process, which is called lazy re-encryption, enables all participants including revoked ones, to be able to see the unchanged file until the next update of a shared file. Therefore, service provider can schedule re-encryption process in a pre-determined time period.

In TT-SFSS, we propose an alternative approach for providing efficient participant revocation for leaving participants. Our novel three-tier design proposes of the following confidentiality mechanisms for the file to be shared. Let $\mathcal{U} = \{U_1, U_2, U_3, \ldots U_N\}$ be the set of registered participants in TT-SFSS and let $U_i \in \mathcal{U}$ be the participant, who wants to share a file with other participants in $\mathcal{U}$. First, each participant in $\mathcal{U}$ executes DCKAP to compute group key, K. U_i encrypts F by using AES-256 with randomly generated File Confidentiality Key (FCK). Then, U_i initializes the access permissions for FCK by encrypting with CP-ABE

in [1] and stores encrypted FCK in a Keybag, which is a directory to store encrypted FCKs of a group. Finally, Keybag is encrypted with AES-256 by using the group key K. Therefore, only the participants in $\mathcal{U}$ can have access permissions for the shared file.

All operations of TT-SFSS are realized by using the following services:

(i) Participant Registration Service
(ii) Group Formation Service
(iii) Group Update Service
(iv) File Update and Confidentiality Service
(v) File Confidentiality Key Update Service

Before explaining the TT-SFSS services in details, we assume that the following general assumptions hold.

Definition 3.12. *We assume that the following definitions, properties, services and mechanisms exist for TT-SFSS:*

- *The communications between TT-SFSS servers and participants are secured by using DCKAP.*
- *A revocation list is used for storing certificates of malicious participants. Let U_i be the participant in file sharing group. The certificate of U_i is added into revocation list in the following cases:*
  - *U_i tries to disrupt the group-key computation.*
  - *U_i tries to gain unauthorized access for shared files.*
  - *U_i refuses to renew the certificate of its public-key after the certificate has expired.*
  - *U_i tries to replace the shared file by some file.*
- *A participant, who starts file sharing is assigned as a file administrator.*
- *A file administrator decides the access permissions of the participants for a shared files. When file administrator leaves file sharing, a randomly selected participant in file sharing group is selected as a new file administrator.*

- *A Majority Rule is used as a decision mechanism in groups such as adding new participant, selecting new file administrator or deciding the group termination. In case of equality, the vote of the group administrator is taken into account.*
- *Classical sharing mechanisms for files and directories such as file/record locking, caching, etc. are not elaborated in TT-SFSS.*
- *If all participants leave the group, then the group is terminated. and the files associated with the group is deleted.*

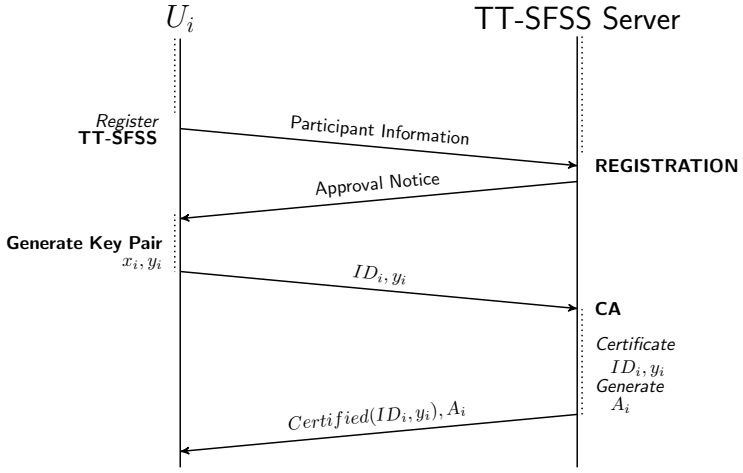

Figure 3.20. Registration and Certification Service

Definition 3.13. *Participant Registration Service. To be able to use the proposed file sharing system, a participant has to register by following the steps shown in Figure 3.20. Let U_i be the new participant that wants to register with the TT-SFSS. First, participant registers by using the cloud interface by entering the necessary information such as username, password and e-mail address. If user is accepted by system, then an approval notice is sent to U_i and user information is stored in the participant database. After the registration, each participant U_i has to generate its long-term key pair for public key certification. The long-term key pair consists of a private key x_i and a public-key $y_i = g^{x_i} mod p$. Unless the public-key of a participant U_i is certified, TT-SFSS does not allow U_i to use file sharing. For certification of public-key, U_i sends (ID_i, y_i) to the CA and CA certificate (ID_i, y_i) as $Certified(ID_i, y_i)$ and generates attribute A_i for U_i.*

To protect system against compromise of a private-key x_i of any participant U_i, the proposed system periodically requests participants to change their key pairs for new public-

key to certify. Participants, who do not change their key pairs are added into the revocation list and accounts of these participants are suspended.

Definition 3.14. *Group Formation Service. When a participant $U_i \in \mathcal{U}$ wants to share a file or files with a group of participants, where $\mathcal{U} = \{U_1, U_2, \ldots, U_N\}$. Group formation service of TT-SFSS operates as defined in 3.21.*

1. U_i selects set of participants $\mathcal{U} = \{U_1, U_2, \ldots, U_n\}$ for sharing file from participant database. Since U_i is the initiator of file sharing, it is assigned as a file administrator.
2. Each participant U_j in $\mathcal{U} \cup U_{TT-SFSS}$ executes ICKAP to compute group key K, where $U_{TT-SFSS}$ is TT-SFSS instance.
3. U_i creates file F and KeyBag KB to be shared with participants in $\mathcal{U}$.
4. U_i randomly generates $256-bit$ key FCK and encrypts F by using AES in CBC mode as $E_{AES_{FCK}}(F)$.
5. U_i selects attribute set $\mathcal{S}$ based on the attributes of participants in $\mathcal{U}$.
6. U_i encrypts FCK with CP-ABE by using the attributes in $\mathcal{S}$ and locate $E_{ABE}(FCK)$ into the KB. After encrypting the $E_{AES}(F)$ and KB with the group key K as $C = E_K(E_{AES}(F)||KB)$, U_1 uploads C to the File System in TT-SFSS Cloud Servers.

Figure 3.21. Group Formation Service

Definition 3.15. *Group Update Service. In TT-SFSS, file sharing groups may change in time. For instance, new participants can join the group, existing participants can leave the group or malicious participant can be extracted from the group. Therefore, TT-SFSS provides Group Update Service. Let $\mathcal{U} = \{U_1, U_2, \ldots, U_n\}$ be the set of participants, $\mathcal{S} = \{S_1, S_2, \ldots, S_n\}$ be the set of attributes for each participant $U_i \in \mathcal{U}$ and $\mathcal{F} = \{F_1, F_2, \ldots, F_r\}$ be the set of shared files, details of the service is as shown in Figure 3.22.*

```
1  if m new participants U_{n+1}, U_{n+2}, U_{n+m} joins then
2      Participants 𝒰 ∪ {U_{n+1}, U_{n+2}, ..., U_{n+m}} ∪ {U_{TT-SFSS}} execute join(·)
       function of DCKAP to update the group key.
3      Let 𝒮' = {S_1, S_2, ..., S_n} ∪ {S_{n+1}, S_{n+2}, ..., S_{n+m}} be the updated set of
       attributes.
4  else
5      Participants 𝒰 ∪ {U_{TT-SFSS}} − {U_{i+1}, U_{i+2}, ..., U_{i+m}} execute leave(·)
       function of DCKAP to update the group key.
6      Let 𝒮' = {S_1, S_2, ..., S_n} − {S_{i+1}, S_{i+2}, ..., S_{i+m}} be the updated set of
       attributes.
7      Add U_{i+1}, U_{i+2}, ... U_{i+m} into the revocation list
8  end
9  foreach file F_j ∈ ℱ do
10     if file administrator of F_j is leaving participant then
11         Assign participant U_l, where l is the smallest index in 𝒰, as a file
           administrator.
12     end
13     File administrator U_k ∈ 𝒰 of F_j generates proxy re-encryption key by using
       CP-ABE and the updated attribute set 𝒮'
14     U_k encrypts proxy re-encryption key with K and uploads to the TT-SFSS
       server.
15     U_{TT-SFSS} updates E_{ABE}(FCK_j) with proxy re-encryption key.
16 end
```

Figure 3.22. Group Update Service

Definition 3.16. *File Update and Confidentiality Service. After the group has been formed and file has been shared as defined in the Group Formation Service, participants in the group $\mathcal{U}$ may update the file F. Let $\mathcal{U} = U_1, U_2, \ldots, U_n$ be the set of participants, then the File Update and Confidentiality Service operates as given in Figure 3.23.*

A participant $U_i \in \mathcal{U}_j$ send encrypted request by using group key K and un-encrypted group identification ID_j for some file F_k from TT-SFSS.
TT-SFSS checks U_i really have an access to F_k by controlling the participant database and trying to decrypt the KeyBag KB_j.
if *U_i have access permission for F_k* **then**
    TT-SFSS sends response that contains $E_K(E_{AES_{FCK}}(F_k)||E_{ABE}(FCK))$
end
Upon receiving $E_K(E_{AES_{FCK}}(F_k)||E_{ABE}(FCK))$, U_i decrypt the response message by using K.
if *attributes of U_i is in $\mathcal{S}$* **then**
    U_i decrypts $E_{ABE}(FCK))$
    By using FCK, U_i decrypts $E_{AES_{FCK}}(F_k)$
    After updating F_k, U_i re-encrypts F_k by using FCK
    Finally, U_i uploads $E_K(E_{AES_{FCK}}(F_k))$ to the TT-SFSS File System.
end

Figure 3.23. File Update and Confidentiality Service

Definition 3.17. *File Confidentiality Key Update Service. In order to protect shared files from revealing of FCK's, TT-SFSS provide service for periodically updating the randomly generated file confidentiality key. Let $\mathcal{U} = \{U_1, U_2, \ldots, U_n\}$ be the group of participants that shares the set of files $\mathcal{F} = \{F_1, F_2, \ldots, F_m\}$. Let $\mathcal{S} = \{S_1, S_2, \ldots, S_n\}$ be the set of attributes for each participant in $\mathcal{U}$ Details of the service is as shown in Figure 3.24.*

```
1  foreach file administrator U_i ∈ 𝒰 do
2      foreach shared file F_j ∈ 𝒰 of U_i do
3          if ((today − LUD) > PUD for any F_j ∈ ℱ) and U_i is then
4              U_i downloads F_j from TT-SFSS server as
               M = E_K(E_{AES_{FCK_j}}(F_k)||E_ABE(FCK_j))
5              U_i decrypts M by using the group key K
6              U_i decrypts E_ABE(FCK_j) by using the CP-ABE and 𝒮
7              U_i decrypts E_{AES_{FCK_j}}(F_j) by using FCK_j
8              U_i randomly selects FCK'_j and encrypts F_j as E_{AES_{FCK'_j}}(F_j)
9              U_i encrypts FCK'_j as E_ABE(FCK'_j)) by using the CP-ABE and 𝒮
10             U_i encrypts and uploads F_j and E_ABE(FCK'_j)) by using the group key K
               as M' = E_K(E_{AES_{FCK'_j}}(F_k)||E_ABE(FCK'_j))
11         end
12     end
13 end
```

Figure 3.24. File Confidentiality Key Update Service

3.6.3. Security of TT-SFSS

In this section we introduce security analysis of TT-SFSS with respect to the following security properties:

(i) File Privacy: Shared files can only be viewed / modified by the group members.

(ii) Escrow-Freeness: No authorized third party may gain access to encryption keys of shared file groups even in emergency

(iii) Forward Secrecy: The forward secrecy property is used for providing security against the compromise of previous and subsequent keys when the participant's long-term private key is compromised.

(iv) Fine Grained Access Control: For this property, we consider the presence of Attribute Based Access Control mechanism.

(v) Fault Tolerance: Property for detecting the malicious participants, who try to disrupt the computation of the group-key.

3.6.3.1. File Privacy. Our primary concern for file privacy is that to protect the content of the shared file against unauthorized attempts of employees of service providers. In the proposed system, TT-SFSS instance is only allowed to join the computation of group key and they are not allowed to access the FCKs unless access permissions are granted by file administrators. Therefore, TT-SFSS provides file privacy. As defined in Section 3.3, any passive attacker $U_j \notin \mathcal{U}$, who is a registered participant of TT-SFSS, tries to access the shared file F of participants in $\mathcal{U}$, cannot compute the group key of $\mathcal{U}$. Therefore, file privacy against passive attacks is also satisfied.

3.6.3.2. Escrow-Freeness. File sharing systems with key escrow mechanisms store the confidentiality keys of shared files in an escrow. Then, for recovery purposes, TTPs may gain access to these confidentiality keys. In TT-SFSS, FCKs of shared files are stored escrow-like structure called KeyBag. However, allowing a TTP to gain access the escrow may have security vulnerabilities such as disclosure of all keys in the escrow or performance overhead due to the disclosed keys must be re-generated and corresponding files have to be re-encrypted. On the other hand, in TT-SFSS, only the set of participants, whose access permissions have been determined by the file administrator, are able to decrypt file confidentiality keys by using their attributes. For instance, if the participant records of a file administrator is deleted by mistake from TT-SFSS participant database, then other participants in file sharing group still have an access for the shared file. Therefore, there is no need for key escrow mechanism in TT-SFSS servers.

3.6.3.3. Fine-Grained Access Control. Another important security property for secure file sharing systems is the fine-grained access control. In file sharing systems, fine-grained access control is satisfied by the existence of attribute based access control, in which the access rights of participants is determined by the use of some policies together with the

use of attributes. In TT-SFSS, we use CP-ABE in [1] to achieve the fine-grained access control. According to the CP-ABE concept, an encrypted data can only be accessed by the participants that have the right attributes for decrypting the encrypted data. In our assumption, the encrypted data is the randomly generated file confidentiality keys. Therefore, let $\mathcal{U} = \{U_1, U_2, \ldots, U_n\}$ be the set of participant for the file sharing group with the set of attributes $\mathcal{S} = \{S_1, S_2, \ldots, S_n\}$. Any participant $U_i \in \mathcal{U}$ can create a smaller group with participants $\mathcal{U}' = \{U_2, U_5, U_7, U_{15}\}$, where $\mathcal{U}' \subset \mathcal{U}$, to share a file F. Let FCK_i be the randomly generated file confidentiality key for file F. By using the set of attributes $\mathcal{S}' = \{S_2, S_5, S_7, S_{15}, S_i\}$, U_i can encrypt the file confidentiality key FCK_i. Then, only the participants with right attributes can obtain the FCK_i.

3.6.3.4. Forward Secrecy. Forward secrecy is not only important protocol for group key agreement protocols but also important for file sharing systems that stores encryption keys in an encrypted directory such as key escrow or Keybag for our case. According to the definition forward secrecy, the property is used for providing security against the compromise of previous and subsequent keys when the participant's long-term private key is compromised. In TT-SFSS, as defined in Section 3.3.1, forward secrecy is satisfied by the existence of temporary public key distribution in DCKAP.

3.6.3.5. Comparative Evaluation of Security. As shown in Table 3.5, we compared TT-SFSS with other protocols in literature with respect to the "Escrow-Freeness", "File Privacy", "Forward Secrecy", and "Fine-Grained Access Control" properties. Comparison results show that only TT-SFSS is able to satisfy all of the criteria. File sharing systems in [74, 90, 104] and WP-IBE also provides almost all of the comparison criteria except forward secrecy. Since these file sharing systems provide escrow-free architecture, they do not need to satisfy forward secrecy property. On the other hand, HABE only concentrate on satisfying hierarchical attribute based encryption. Mona and pNFS-AKE I are file sharing systems, which only provides file privacy. In addition to TT-SFSS, pNFS-AKE III also provides forward secrecy in parallel network file systems. Moreover, FADE in [70], TimePRE in [75], FH-CP-ABE in [105], the systems in [74, 90, 106] are other examples for the use of fine-grained access control in secure file sharing systems. In addition to Mona and pNFS-

AKE I, FADE in [70], WP-IBE in [107], FH-CP-ABE in [105], the schemes in [67, 108] are good examples for file privacy property.

Table 3.5. Comparison of Secure File Sharing Systems

File Sharing Systems	**Escrow-Freeness**	**File Privacy**	**Forward Secrecy**	**Fine-Grained Access Control**
System in [67]	✓	✓	✗	✗
pNFS-AKE I in [71]	✗	✓	✗	✗
pNFS-AKE II in [71]	✓	✓	✗	✗
pNFS-AKE III in [71]	✓	✓	✗	✗
HABE in [79]	✗	✗	✗	✓
Mona in [68]	✗	✓	✗	✗
System in [74]	✓	✓	✗	✓
System in [108]	✗	✓	✗	✗
FADE in [70]	✗	✓	✗	✓
TimePRE in [75]	✗	✗	✗	✓
System in [90]	✓	✓	✗	✓
WP-IBE in [107]	✓	✓	✗	✗
System in [106]	✓	✗	✗	✓
FH-CP-ABE in [105]	✗	✓	✗	✓
System in [109]	✓	✓	✗	✗
System in [104]	✓	✓	✗	✓
TT-SFSS	✓	✓	✓	✓

3.6.4. Comments on the Applicability of TT-SFSS

In this section, we analyze the applicability of TT-SFSS with respect to comparisons of the primary level revocation and participant revocation in Attribute Based Access Control (ABAC). For ABAC, we use CP-ABE since it is one of the most preferred ABAC mechanism in file sharing systems [75, 79, 90, 105, 106]. All simulations of TT-SFSS and CP-ABE are carried out by using Python 2.7 and Charm Framework [103], pyPEBEL and pyCrypto that

simulations are run on a DELL PC with 500 GB HDD disk, 3.2 GHz Intel Core i5 processor and 16GB 1600 MHz DDR3 RAM.

Table 3.6. Performance Comparison for Participant Revocation and Re-Encryption with CP-ABE

File Size	**KeyBag Encryption with AES + Group Key**	**File Re-Encryption CP-ABE**
1 GB	14272 msec	35 min 49 sec
500 MB	8769 msec	17 min 51 sec
100 MB	3785 msec	3 min 36 sec
50 MB	3181 msec	1 min 42 sec
10 MB	2635 msec	20 sec

In Table 3.6, we have compared the performance of primary revocation of TT-SFSS and re-encryption cost by using CP-ABE. Simulation results show that TT-SFSS provides better performance than re-encryption operation by using CP-ABE. Although ABE-variants provide fine-grained access control, they cannot encrypt files larger than 100 MB in considerable time. On the other hand, we use three-tier encryption approach to use fine grained access control property of CP-ABE and fast processing capabilities of AES. Moreover, simulation results show TT-SFSS provides better performance for participant revocation. Since the use of CP-ABE is only restricted to encrypting/decrypting 256-bit FCK keys, TT-SFSS also provides better performance in general.

3.7. Discussions

In this chapter, we proposed an improved conference-key agreement protocol for dynamic groups, called DCKAP. The proposed protocol uses a modified form of Tseng's protocol as ICKAP and has new ACKA operations to provide dynamic group management. We analyzed proposed protocol regarding the security and performance. The analysis show that DCKAP withstand the known passive and active attacks for conference-key agreement protocols such as eavesdropping and impersonation. In addition, DCKAP preserves correct-

ness, fault-tolerance and forward secrecy properties. Furthermore, the protocol is also secure against known-key attack scenario for ACKA operations. Thus, DCKAP has better fault correction with respect to Tseng's protocol. To investigate the additional application areas of DCKAP, we also present comparative scalability analysis of DCKAP with other dynamic group key agreement protocols. The scalability analysis has shown that DCKAP can also be applicable as efficient participant revocation mechanism for secure file sharing systems. Therefore, we proposed a novel file sharing system called TT-SFSS. TT-SFSS provides efficient participant revocation. Moreover, TT-SFSS also provides basic security properties such as escrow-freeness, forward secrecy, fine-grained access control and file privacy. Simulation results show that TT-SFSS provides better performance for participant revocation than re-encryption by using CP-ABE.

4. KAP-PBC: KEY AGREEMENT PROTOCOL WITH PARTIAL BACKWARD CONFIDENTIALITY

Group key agreement protocols are one of the best candidate for establishing a secure communication in a distributed network. The early designs of group key agreement protocols focus on static groups, in which the set of participants does not change until the end of a communication session. However, with the growth of technology, dynamic group structures replaced the static groups in multi-party communications. Group key agreement protocols that were designed for static groups became outdated since handling of updating a group key has a challenging overhead. Therefore, group key agreement protocols are evolved to overcome this overhead by providing dynamic group operations. Dynamic group key agreement protocols such as [16, 93, 102, 110] update the group key with performing less effort than static group based key agreement protocols. Moreover, group key agreement protocols with dynamic group capability are achieving the basic security properties with the static ones.

In general, group key agreement protocols are based on Diffie-Hellman protocol [2], which enables only two participants to agree on a common key. Later, the first multi-participant group key agreement protocol was proposed in [3]. In addition, there have been many studies about group key agreement protocols with different security properties. One well-known property of group key agreement protocols is the authentication, which is used for confirming the identities of participants in the group communication [8]. Two important group key agreement protocols with and without authentication were proposed by Burmester and Desmedt in [7]. Both of these protocols are for static groups not for dynamic groups. Authentication with anonymity may be used as a security property in various application areas, such as IoT-enabled devices in a distributed cloud computing environment [111], patient monitoring system using wireless medical sensors [112] and payment systems [113].

There are two additional security properties for dynamic group key agreement protocols, namely backward confidentiality and forward confidentiality [16, 18]. In backward confidentiality, participants who joined the group cannot compute former group keys. In for-

ward confidentiality, participants who left the group cannot compute subsequent group keys. Dynamic group key agreement protocols are expected to be used in applications such as teleconferences, instant communications, file sharing systems, etc. On the other hand, there exist a number of problems on the use of existing dynamic group key agreement protocols in File Sharing Systems (FSS) such as lack of privacy [71], violation of availability [67] and dependency for key escrow [70]. The most important reason of problems in FSS is the existence of backward confidentiality property. Since joining participants cannot compute the previous group key just before joining the group, FSS must provide a mechanism to grant access permissions for joining participants. Trusted third parties (TTPs) or dedicated participants in the group (for instance Group Managers) are used to overcome this problem. However, if TTPs are involved in file sharing, the privacy of the file is endangered. If dedicated participants distributes group key, there is a possibility for the violation of availability due to the single-point-of-failure. Moreover, if TTPs and dedicated participants exist in file sharing systems, the key escrow mechanism provides data recovery keys for encrypted files [20]. Since files are shared by the participants of a communication group, there is no need for such backup mechanism. In this chapter, our main motivation is to solve these problems with the provision of partial backward confidentiality.

Our contributions for this chapter are given below:

(i) We propose a new security property called the Partial Backward Confidentiality (PBC). In PBC, a new participant can compute the last valid group key just before joining the group but the new participant cannot compute former group keys.

(ii) Moreover, we propose a Key Agreement Protocol with Partial Backward Confidentiality (KAP-PBC). KAP-PBC design is based on the protocol in [15] to provide operations for dynamic groups while preserving the basic security properties.

(iii) We also present a proof of concept case study called Private File Sharing System (PFSS) to demonstrate the applicability of the partial backward confidentiality property to solve the lack of privacy, the violation of availability and the dependency for key-escrow problems.

The rest of the chapter is organized as follows. The comparison of the dynamic group key agreement protocols with respect to are given in the next section. KAP-PBC is proposed in Section 4.2. Performance analysis and security analysis are given in Section 4.3 and 4.4, respectively. Finally, Section 4.5 concludes the chapter.

4.1. Comparison of Dynamic Group Key Agreement Protocols with respect to Protocol Properties

In this section, we give the comparison of previously proposed dynamic group key agreement protocols and KAP-PBC as shown in Table 4.1. Criteria used for comparing protocol properties are listed as follows:

(i) Dynamic Group Operations (DGO): Dynamic group operations can be listed as join, leave, mass join (merge) and mass leave (divide).

(ii) Security Properties for Group Key Agreement Protocols (SPGKAP): The basic parameters to assess the security level of a group key agreement protocols are authentication, fault-tolerance and forward secrecy.

(iii) Security Properties for Dynamic Group Key Agreement Protocols (SPDGKAP): In group key agreement protocols, security of the resulting group key after dynamic group operations can be assessed by the existence of Backward Confidentiality and Forward Confidentiality properties.

(iv) Partial Backward Confidentiality (PBC): The last criterion for dynamic group key agreement protocols is the Partial Backward Confidentiality property. With this property, a new participant can compute the last valid group key just before joining the group but the new participant cannot compute former group keys.

Table 4.1. Comparison of Protocol Properties

Protocols	**DGO**				**SPGKAPs**			**SPDGKAPs**		**PBC**
	Join	Leave	Mass Join	Mass Leave	Authenti-cation	Fault-Tolerance	Forward Secrecy	Backward Conf.	Forward Conf.	
Ermis *et al.* [16]	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗
Foss [93]	✓	✓	✗	✗	✓	✗	✗	✓	✓	✗
Shi *et al.* [11]	✓	✗	✗	✗	✓	✓	✗	✓	✗	✗
Tseng [17]	✓	✓	✗	✗	✓	✓	✓	✗	✗	✗
Cheng *et al.* [98]	✓	✓	✗	✗	✓	✓	✗	✓	✓	✗
Chung [5]	✓	✓	✗	✗	✓	✗	✓	✓	✓	✗
Kim *et al.* [114]	✓	✓	✗	✗	✓	✗	✗	✓	✓	✗
Hao *et al.* [115]	✓	✓	✗	✗	✓	✗	✗	✓	✓	✗
Xiaozhuo *et al.* [116]	✓	✓	✗	✗	✓	✗	✗	✓	✓	✗
Mortazavi *et al.* [117]	✓	✓	✗	✗	✗	✗	✗	✗	✗	✗
Zhang *et al.* [118]	✓	✓	✗	✗	✓	✓	✓	✓	✓	✗
Zhao *et al.* [110]	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗
Zhang *et al.* [102]	✓	✓	✗	✗	✓	✓	✓	✓	✓	✗
Xia *et al.* [119]	✓	✓	✓	✗	✓	✓	✓	✓	✓	✗
KAP-PBC	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

As seen in Table 4.1, we have compared the proposed protocol with other protocols in the literature regarding the dynamic group operations, security properties of group key agreement protocols, dynamic security properties of group key agreement protocols and partial backward confidentiality property. Since it is firstly proposed in this thesis, the only protocol that provides partial backward confidentiality is KAP-PBC. Protocols in [16, 110] and KAP-PBC satisfy all of the criteria that a dynamic group key agreement protocol can. Moreover, [16, 110] and KAP-PBC have extra operations for efficiently handling of mass join and mass leave operations. Specifically, if a protocol provides mass join and mass leave operations, then a protocol can accomplish join or leave operation at one execution instead of executing separate operations for each join or leave. Therefore, the performance of protocols that provide mass join and mass leave operations is better than protocols that provide single join and leave operations.

In terms of security criteria, we have compared protocols with respect to SPGKAPs and SPDGKAPs. Protocol in [117] has the worst protocol among other protocols since it does not satisfy any of the security criteria. In addition, protocols in [114–116] do not provide neither fault-tolerance nor forward secrecy. Therefore, these protocols are vulnerable against security threats such as malicious attempts to compute a wrong group key or compromise of group keys. On the other hand, protocol in [17] does not provide forward and backward confidentiality properties, which causes the protocol to expose former or subsequent group keys after the set of participant is altered. When forward confidentiality property or backward confidentiality property is not provided, joining participants or leaving participants can view former or subsequent communications in the group.

4.2. A Key Agreement Protocol with Partial Backward Confidentiality

This section presents the Key Agreement Protocol with Partial Backward Confidentiality, namely KAP-PBC. The protocol is based on the static conference key agreement protocol in [15], which extends [9] with forward secrecy and provides better performance than protocols in [6, 14]. For conference key agreement protocols, static groups are mostly used since they operate for a fixed time period and they have a fixed set of participants [6, 7, 14]. On the contrary, for file sharing systems, there is no such time and a set of restriction ex-

ist. Therefore, we propose KAP-PBC to be able to use group key agreement protocols in file sharing systems. Improvements in KAP-PBC help operate on dynamic groups and provide a new security property called Partial Backward Confidentiality. For providing dynamic group operations, secret-key generation of [15] is modified. In order to increase the resistance of KAP-PBC against impersonation attacks, we have modified the signatures in [15] with Schnorr signature scheme. Moreover, we define join with partial backward confidentiality operation to solve the lack of privacy, violation of availability and dependency for key-escrow problems of file sharing systems.

In Figure 4.1, the general organization of KAP-PBC is shown as a diagram. First, any registered participant U_i selects the set of registered participants to form a communication group as $\mathcal{U} = \{U_1, U_2, \ldots, U_n\}$. Next, U_i invites participants in $\mathcal{U}$ by using $invite(\cdot)$ function. Participant, who invites other participants for KAP-PBC execution, is selected as an administrator. Each participant $U_j \in \mathcal{U}$, who accepts the invitation of U_i executes $sendPublic\text{-}Key(\cdot)$ function and broadcasts temporary public keys. Then, each participant $U_j \in \mathcal{U}$ verifies received public keys of other participants in the group by using $verifyPublicKey(\cdot)$ function. If any fault is detected, $correct(\cdot)$ function is executed to remove malicious participants. After malicious participants are removed from the set of participants, each participant $U_j \in \mathcal{U}$ consecutively executes $sendSecretKey(\cdot)$ to broadcast the secret key and $verifySecretKey(\cdot)$ to verify received secret keys of other participants. If any malicious participant is detected while verifying the secret keys, then $correct(\cdot)$ is executed for removing malicious participants. After all malicious participants are removed from the set of participants, the rest of the participants compute a group key by executing $compute(\cdot)$ function. Once the group key is computed, each participant U_j periodically checks if any of "join", "join with partial backward confidentiality", "leave", "key update" or "terminate group communication" is occurred or not. Then, each participant U_i executes corresponding $join(\cdot)$, $joinPBC(\cdot)$, $leave(\cdot)$ or $rekey(\cdot)$ function as defined in figures.

Let $\mathcal{U} = \{U_1, U_2, \ldots, U_n\}$ be the selected set of participants from $\{U_1, U_2, \ldots, U_m\}$, where $m >> n$, and let any $U_i \in \mathcal{U}$ be the group administrator that initiates KAP-PBC execution. Then, KAP-PBC is defined as follows:

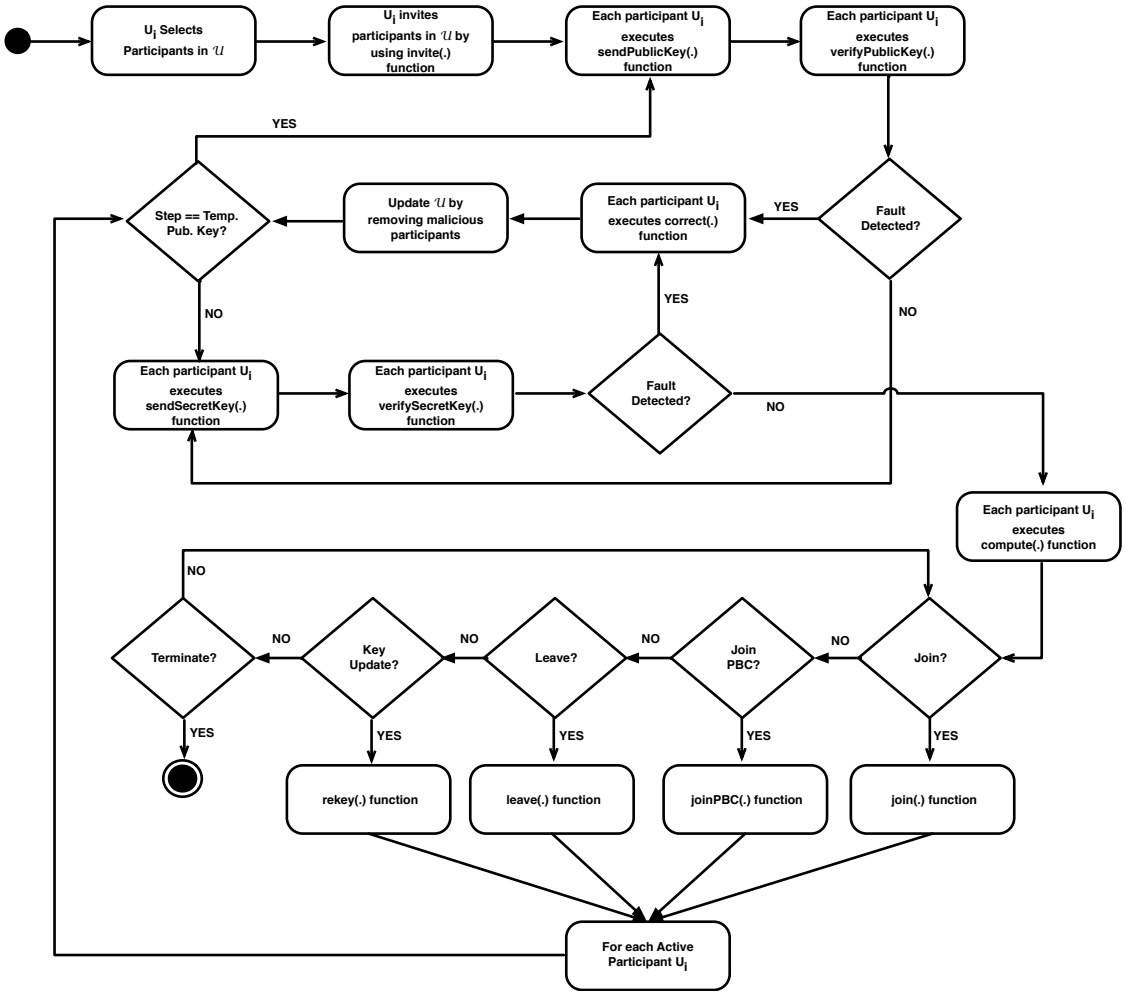

Figure 4.1. Diagram of KAP-PBC

Definition 4.1. *Function Call.* $U_i \rightarrow generic(\cdot)$ *means that group participant* U_i *performs function* $generic(\cdot)$.

Definition 4.2. *Administrator Participant. In KAP-PBC, we assume that the following cases are hold for administrator participant:*

- *All participants in* $\mathcal{U}$ *including the administrator participant have the same computation power.*
- *The group key is updated when a new participant joins or an existing participant leaves the group. On the other hand, if communication group is not updated for a period of time, the administrator participant randomly selects a participant in the group for updating the group key. This period of time is denoted as* $key-update-period$ *in KAP-PBC.*

- *When new participant joins the group a Majority Rule is used as a decision mechanism. In case of equality, the vote of the group administrator is taken into account.*
- *When administrator participant U_i leaves the group, a participant $U_j \in \mathcal{U}$ is selected as a new administrator participant, where j is the smallest index in $1,\ldots,n$ and $i \neq j$.*

Definition 4.3. *The KAP-PBC protocol operates as given in the Figure 4.2.*

```
U_i initializes the set of participants group 𝒰 = {U_1, U_2, U_3, ..., U_n}
U_i → invite(·)
U_i → key-update-period
U_j ∈ 𝒰 → sendPublicKey(·)
U_j ∈ 𝒰 → verifyPublicKey(·)
U_j ∈ 𝒰 → sendSecretKey(·)
U_j ∈ 𝒰 → verifySecretKey(·)
while ∃U_t ∈ 𝒰, where V_{j,t} = "failure" for j ≠ t do
    U_l → correct(·), where l ≠ j ≠ t
end
U_j ∈ 𝒰 → compute(·);
while True do
    if 𝒰 changes in key-update-period then
        U_j ∈ 𝒰 → join(·) (joinPBC(·) if necessary) U_j ∈ 𝒰 → leave(·)
    else
        U_j ∈ 𝒰 → rekey(·)
    end
end
Terminate Group.
```

Figure 4.2. KAP-PBC Function

Definition 4.4. *Invitation Function, $invite(\cdot)$. The administrator participant, U_i, invites the other participants in the set of participants by sending the group information and its ID. The algoritm is given in Figure 4.3.*

```
forall the U_j ∈ 𝒰 − {U_i}: do
    broadcast invitation message M : (𝒰, ID_i)
end
```

Figure 4.3. Invitation Function, $invite(\cdot)$

Definition 4.5. *Temporary Public-Key Distribution Function, sendPublicKey*$(\cdot)$. *After the invitation message has been sent, each participant* $U_i \in \mathcal{U}$ *calculates temporary public key for providing more secure group key computation as shown in Figure 4.4.*

```
Randomly select t_i ∈ Z_q^*
ω_i = g^{t_i} mod p
e_{1,i}, s_{1,i} = SS(y_i, x_i, ω_i)
Broadcast the message M_{1,i} = (ω_i, e_{1,i}, s_{1,i}, T).
```

Figure 4.4. Send Public Keys, $sendPublicKey(\cdot)$

Definition 4.6. *Verification of Temporary Public-Key Function, verifyPublicKey*$(\cdot)$. *In KAP-PBC, after each message exchange, each participant* U_i *verifies the incomming messages from other participants in the group. Verification of temporary public-key function is as given in Figure 4.5:*

```
forall the U_j ∈ 𝒰 and j ≠ i do
    U_i checks ω_j is really issued by U_j by using SV(y_j, e_{1,j}, s_{1,j}, ω_j)
end
```

Figure 4.5. Verification of Temporary Public Keys, $verifyPublicKey(\cdot)$

Definition 4.7. *Secret Key Distribution Function, sendSecretKey*$(\cdot)$. *This function is used to distribute the secret-key to the other participants in the group. Each participant* $U_i \in \mathcal{U}$ *executes the sendSecretKey*$(\cdot)$ *function as showin in Figure 4.6:*

U_i randomly selects an integer $a_i \in Z_q^*$

U_i calculates $k_{ij} = (\omega_j^{a_i} mod p) mod q, where 1 \leq j \leq n$ and $i \neq j$

U_i randomly selects a line $L(x)$, $L(x) = c_i x + CK_i mod q$, where $c_i = g^{a_i} mod p$. U_i computes the sub-key CK_i as shown in below:

if $i+1 \leq n$ **then**

  $CK_i = \omega_{(i+1)}^{t_i} mod p = g^{t_i t_{i+1}} mod p$

else

  $CK_i = \omega_{(i+1) mod n}^{t_i} mod p = g^{t_i t_{(i+1) mod n}} mod p$

U_i calculates the values $d_{i,j} = L(k_{i,j}) mod q, where 1 \leq j \leq n$

U_i calculates $d'_{i,j} = k_{i,j} \oplus d_{i,j}$, where $1 \leq j \leq n$ $e_{2,i}, s_{2,i} = SS(y_i, x_i, CK_i)$

U_i broadcasts the message $M_{2,i} = \{T, e_{2,i}, s_{2,i}, c_i, d'_{i,1}, d'_{i,2}, \ldots, d'_{i,n}\}$

Figure 4.6. Secret Key Distribution Function, $sendSecretKey(\cdot)$

Definition 4.8. *Verification of Secret Keys Function, $verifySecretKey(\cdot)$. Each participant $U_i \in \mathcal{U}$ verifies the shared secret-keys as given in Figure 4.7.*

forall the $U_j \in \mathcal{U} - U_i$ **do**

U_i recovers the sub-key CK_j

U_i checks the time-stamp T is valid or not

U_i calculates the common session key $k_{j,i}$ by using

$$k_{j,i} = (c_j^{t_i} mod p) mod q \tag{4.1}$$

Then, U_i calculates the subkey CK_j by using the following equations:

$$d_{j,i} = d'_{j,i} \oplus k_{j,i} \tag{4.2}$$

$$CK_j = d_{j,i} - c_j k_{j,i} mod q \tag{4.3}$$

if $SV(y_j, e_{2,j}, s_{2,j}, CK_j)$ *holds* **then**

$V_{i,j} =$ "*success*"

else

$V_{i,j} =$ "*failure*"

end

end

Figure 4.7. Verification of Secret Keys Function, $verifySecretKey(\cdot)$

Definition 4.9. *Fault Correction Function, $correct(\cdot)$. After the verification of broadcast messages in $verifyPublicKey(\cdot)$ or $verifySecret(\cdot)$ functions, if potential malicious participants exist in the V, then $U_i \in \mathcal{U}$ and $U_i \neq U_j \neq U_k$ executes the function in Figure 4.8.*

```
forall the U_k ∈ 𝒰, where V_{j,k} = "failure" do
    if re-verification of U_k by U_i fails then
        U_k is removed from the participant list, 𝒰 ← 𝒰 − {U_k}
    else
        U_j is removed from the participant list, 𝒰 ← 𝒰 − {U_j}
end
```

Figure 4.8. Fault Correction Function, $correct(\cdot)$

Definition 4.10. *Key Computation Function, $compute(\cdot)$. If no fault is detected during the verification of temporary public keys and secret keys, each participant $U_i \in \mathcal{U}$ computes the group key as shown in Figure 4.9.*

Each participant by $U_i \in \mathcal{U}$ computes the group key as shown in the following equations:

$$\begin{aligned} CK &= ((CK'_1 CK'_2 \cdots CK'_m) mod\, p) mod\, q \\ &= (g^{t_1 t_2 + t_2 t_3 + \ldots + t_{n-1} t_n + t_n t_1} mod\, p) mod\, q \end{aligned}$$

Figure 4.9. Key Computation Function, $compute(\cdot)$

Definition 4.11. *Join Function, $join(\cdot)$. Participant join function is used to handle the mass and single join of new participants into the previously established group communication. Let $m \geqslant 1$ be the number of joining participants for the participant set $\mathcal{U} = \{U_1, U_2, \ldots, U_n\}$. Procedure for join operation is as shown in Figure 4.10.*

```
if U_i ∈ {U_1, U_2, ..., U_{n-1}} then
    U_i broadcasts the previously calculated messages in
    sendPublicKey(·) and sendSecretKey(·) functions of KAP-PBC
end
if U_i ∈ {U_n, U_{n+1}, ..., U_{n+m}} then
    U_i → sendPublicKey(·) if no fault occurs during the execution then
        U_{n-1} → sendSecretKey(·)
    end
end
if ∀V_{i,j} = "success", where i, j ∈ {1, 2, ..., n+m} and i ≠ j then
    forall the U_i ∈ {U_1, U_2, ..., U_{n+m}} do
        U_i → compute(·)
    end
end
```

Figure 4.10. Join Function, $join(\cdot)$

Definition 4.12. *Join with PBC Function, joinPBC*$(\cdot)$. *Partial backward confidentiality allows a new participant to compute the last valid group key just before joining the group but not to compute former group keys. Let* $\mathcal{U}' = \{U_{n+1}, U_{n+2}, \ldots, U_m\}$ *be the set of joining participants at time* T_1 *for* $\mathcal{U} = \{U_1, U_2, \ldots, U_n\}$ *with the group key:*

$$K = (g^{t_1t_2+t_2t_3+\cdots+t_{n-1}t_n+t_nt_1} mod\, p) mod\, q$$

Then the procedure for join operation with partial backward confidentiality at time T_2 *for each participant* $U_i \in \mathcal{U} \cup \{U_{n+1}\}$ *is as shown in Figure 4.11.*

if $U_i \in \{U_1, U_2, \ldots, U_{n-1}\}$ **then**
  U_i broadcasts the previously calculated messages by using $sendSecretKey(\cdot)$ function
else if $U_i == U_n$ **then**
  $U_n \rightarrow sendSecretKey(\cdot)$ and uses the computed secret in the previous execution of KAP-PBC, $g^{t_n t_1}$, as the temporary public key
else
  $U_{n+1} \rightarrow sendPublicKey(\cdot)$
if *No fault occurs during the execution* **then**
  U_{n+1} computes $g^{t_n t_1}$ from the $sendSecretKey(\cdot)$ output of U_n, $g^{t_n t_1 t_{n+1}}$
  $U_{n+1} \rightarrow compute(\cdot)$ for computing K by using the outputs of $sendSecretKey(\cdot)$ participants in $\{U_1, U_2, \ldots, U_n\}$ as follows:

$$K = (g^{t_1 t_2 + t_2 t_3 + \cdots + t_{n-1} t_n + t_n t_1} mod\, p) mod\, q$$

end

Figure 4.11. Join with PBC Function, $joinPBC(\cdot)$

The following cases exist for join operation with partial backward confidentiality to be used later in security analysis:

- As seen in the definition, independent from the number of joining participants, *joinPBC*$(\cdot)$ function is only executed for one new joining participant and the existing participants in the group.
- During the execution of *joinPBC*$(\cdot)$, if any malicious attempt of any participant is detected, then *correct*$(\cdot)$ function is executed and the malicious participant is excluded from the group key computation.
- Since it violates the backward confidentiality property, *joinPBC*$(\cdot)$ function cannot be executed consecutively. If necessary, there must be a *rekey*$(\cdot)$ execution between two *joinPBC*$(\cdot)$ function executions.

- If more than one individual users try to join the group at the same time, KAP-PBC processes join as mass join and only the participant, who has the smallest id can compute $joinPBC(\cdot)$ function. For instance, let U_k, U_l, U_m, U_p be the joining participants for $\mathcal{U} = \{U_1, U_2, \ldots, U_n\}$, where $k > m > l > p > n$. Then, U_k, U_l, U_m, U_p executes $join(\cdot)$ function and only U_p executes $joinPBC(\cdot)$ function.

Definition 4.13. *Leave Function, $leave(\cdot)$. If the set $\mathcal{U}' = \{U_i, U_{i+1} \ldots, U_{i+m}\}$ be the set of leaving participants, for $\mathcal{U}' \subseteq \mathcal{U}$ and $|\mathcal{U}'| \geqslant 1$. Then the $leave(\cdot)$ function is as shown in Figure 4.12.*

```
if |U| − |U'| < 2 then
    cancel the group key computation
end
for each leaving participant U_j ∈ U' do
    the non-leaving participant(s) U_{j−1} ∈ U − U', re-executes KAP-PBC protocol
    from sendPublicKey(·) function if no fault occurs then
        each U_{j−1} and U_{j−2} ∈ U − U' → sendSecret(·)
    end
    U_i ∈ U − U', calculates the group key by using new secret-key values
end
```

Figure 4.12. Leave Function, $leave(\cdot)$

Definition 4.14. *Key Update Function, $rekey(\cdot)$. After the secure group communication is established, the group key must be updated. This update can be handled by join and leave operations. However, if no modification occurs in the participant list the key stays without refreshed. Therefore, the $rekey(\cdot)$ function is used to update the key periodically. The definition of the function is as follows:*

1 Administrator of group communication randomly picks a user, $U_i \in \mathcal{U}$
2 $U_i \rightarrow sendPublicKey(\cdot)$
3 $U_{i-1} \rightarrow sendSecretKey(\cdot)$
4 All of the pariticipants executes $compute(\cdot)$ function

Figure 4.13. Key Update Function, $rekey(\cdot)$

4.3. Security Analysis of KAP-PBC

In this section, we give the security analysis of KAP-PBC. First, we show that the proposed protocol has the same security level with existing protocols in literature. Then, we show that KAP-PBC is secure against impersonation, eavesdropping and replay attacks as defined in [15]. Moreover, we prove that KAP-PBC provides backward confidentiality and forward confidentiality properties for join and leave operations. Finally, we show that the partial backward confidentiality is satisfied by KAP-PBC.

4.3.1. Authentication, Fault Tolerance and Forward Secrecy

In this part of the security analysis, we show that KAP-PBC protocol satisfies basic security properties such as authentication, fault tolerance and forward secrecy.

The authentication property in group key agreement protocols is used for detecting whether the participant is a member of the group or not. This detection process is generally realized during the verification of signatures that are appended to the broadcast messages of participants. In KAP-PBC, we use Schnorr signature for signing the broadcast messages in $sendPublicKey(\cdot)$ and $sendSecretKey(\cdot)$ functions and these messages are verified in $verifyPublicKey(\cdot)$ and $verifySecretKey(\cdot)$, respectively. Moreover, while signing the messages, long-term key pairs are used. In KAP-PBC, public keys of participants have to be certificated by CA. Hence, KAP-PBC protocol provides authentication property.

Fault-tolerance property is used for eliminating malicious participants from the group communication. Fault-tolerance property consists of two actions; fault detection and fault correction. In KAP-PBC protocol, faults detection is realized in $verifyPublicKey(\cdot)$ and $verifySecretKey(\cdot)$ functions and possible malicious participants are marked by using the verification matrix, V. After the detection of faults, fault correction is realized by executing the $correct(\cdot)$ function. In this function, verification functions are re-executed. Then, malicious participant is excluded from the group.

The last important security property of group key agreement protocols is that the forward secrecy property. By using forward secrecy, a security mechanism is provided to protect the group key against compromises of produced group keys, such as the compromise of participant's long-term private-key. In KAP-PBC, the long-term private keys are only used in the execution of the Schnorr signature outputs of $sendPublicKey(\cdot)$ and $sendPrivateKey(\cdot)$ functions. Moreover, each participant generates its own temporary public-key in $sendPublicKey(\cdot)$ function. If any participant's temporary private key compromises, then only the group key that these temporary keys are used will be compromised. Hence the KAP-PBC satisfies the forward secrecy property.

4.3.2. Security of KAP-PBC against Impersonation, Eavesdropping and Replay Attacks

The protocol proposed in [15] is secure against well known attacks for group key agreement protocols in literature. Since KAP-PBC is the improved version of [15], we show that the improved version has no security vulnerabilities regarding impersonation, eavesdropping and replay attacks. Let $\mathcal{U} = \{U_1, U_2, \ldots, U_n\}$ be the set of participants, security analysis of KAP-PBC regarding impersonation, eavesdropping and replay attacks is as follows:

First, we analyze the security of KAP-PBC with respect to impersonation attack. In impersonation attacks, an impersonator tries to generate the signature of any participant U_i without knowing the long-term or temporary private keys. In KAP-PBC, a participant can communicate with other participants while executing $sendPublicKey(\cdot)$ function and $sendSecretKey(\cdot)$ function. For each participant U_i, signature parameters are $e_{1,i}$, $s_{1,i}$ and

$e_{2,i}, s_{2,i}$ for broadcast messages $M_{1,i}$ and $M_{2,i}$, respectively. Schnorr signature is secure against impersonation attacks [120]. Moreover, Schnorr signature is also provide security against Related Key Attacks as defined in [121].

Second, we analyze the security of KAP-PBC with respect to eavesdropping attack. Let E be an eavesdropper that tries to obtain information by observing the communications among participants. Then, she can extract information from either *send- PublicKey*$(\cdot)$ function or *sendSecretKey*$(\cdot)$ function. For *send- PublicKey*$(\cdot)$, E has to extract t_i by using $\omega_i = g^{t_i} mod p$, which is hard under the assumption of discrete logarithm problem [17]. Therefore, E can not learn any information about t_i. For security of *sendSecretKey*$(\cdot)$ we use the assumption of Variant Decisional Diffie-Hellman Problem in [6]

Definition 4.15. *Variant Decisional Diffie-Hellman Problem [6]. For $p = 2q+1$, where p and q are large prime numbers. Let G_q be quadratic-residue subgroup of Z_p^*. Given any generators $y_1, y_2 \in G_q - \{1\}$, the following two random variable tuples are computationally indistinguishable:*

$$(y_1, y_2, y_1^R mod p mod q, y_2^R mod p mod q)$$

and

$$(y_1, y_2, u_1, u_2)$$

where $R, u_1, u_2 \in Z_q$.

For *sendSecretKey*$(\cdot)$, the broadcast message for any participant U_i is as follows:

$$M_{2,i} = \{T, e_{2,i}, s_{2,i}, c_i, d'_{i,1}, d'_{i,2}, \ldots, d'_{i,n}\}$$

Let S be the simulator for E. S simulates *sendSecretKey*$(\cdot)$ function for participant U_i as given in 4.14.

S randomly selects ω'_j, t'_i and x'_i
S computes $y'_i = g^{x'_i} mod p$
S randomly selects an integer $a'_i \in Z_q^*$
S calculates $k'_{ij} = (\omega_j'^{a'_i} mod p) mod q$, where $1 \leq j \leq n$ and $i \neq j$
U_i randomly selects a line $L(x)$, $L(x) = c'_i x + CK'_i mod q$, where $c'_i = g^{a'_i} mod p$. S computes the sub-key CK'_i:
if $i+1 \leq n$ **then**
  $CK'_i = \omega_{(i+1)}'^{t'_i} mod p = g^{t'_i t'_{i+1}} mod p$
end
else
  $CK'_i = \omega_{(i+1) mod n}'^{t'_i} mod p = g^{t'_i t'_{(i+1) mod n}} mod p$
end
S calculates the values $dd_{i,j} = L(k_{i,j}) mod q$, where $1 \leq j \leq n$
S calculates $d''_{i,j} = k_{i,j} \oplus dd_{i,j}$, where $1 \leq j \leq n$ $e'_{2,i}, s'_{2,i} = SS(y'_i, x'_i, CK'_i)$
U_i broadcasts the message
$M'_{2,i} = \{T, e'_{2,i}, s'_{2,i}, c'_i, d''_{i,1}, d''_{i,2}, \ldots, d''_{i,n}\}$

Figure 4.14. Simulated Secret Key Distribution Function, $simSendSecretKey(\cdot)$

We assume that the original transcript $M_{2,i}$

$$M_{2,i} = \{T, e_{2,i}, s_{2,i}, c_i, d'_{i,1}, d'_{i,2}, \ldots, d'_{i,n}\} \tag{4.4}$$

and simulated transcript $M'_{2,i}$

$$M'_{2,i} = \{T, e'_{2,i}, s'_{2,i}, c'_i, d''_{i,1}, d''_{i,2}, \ldots, d''_{i,n}\} \tag{4.5}$$

are computationally indistinguishable for any $e_0, s_0 \in Z_q^*$:

$$Pr[e_{2,i} = e_0, s_{2,i} = s_0] - Pr[e'_{2,i} = e_0, s'_{2,i} = s_0] > \frac{1}{q^2} \tag{4.6}$$

Therefore, we only consider the following probability distributions $Pr[(d'_{i,1}, d'_{i,2}, \ldots, d'_{i,n}, c_i)|e_{2,i} = e_0, s_{2,i} = s_0]$ and $Pr[(d''_{i,1}, d''_{i,2}, \ldots, d''_{i,n}, c'_i)|e'_{2,i} = e_0, s'_{2,i} = s_0]$. For any fixed $e_0, s_0 \in Z_q^*$, $CK_i = ck_0$ is also fixed. Then,

$$Pr[(d'_{i,1}, d'_{i,2}, \ldots, d'_{i,n}, c_i)|e_{2,i} = e_0, s_{2,i} = s_0] = Pr[(d'_{i,1}, d'_{i,2}, \ldots, d'_{i,n}, c_i)|CK_i = ck_0] \quad (4.7)$$

and

$$Pr[(d''_{i,1}, d''_{i,2}, \ldots, d''_{i,n}, c'_i)|e'_{2,i} = e_0, s'_{2,i} = s_0] = Pr[(d''_{i,1}, d''_{i,2}, \ldots, d''_{i,n}, c'_i)] \quad (4.8)$$

Theorem 4.1. *Under the assumption of Variant Decision Diffie-Hellman, for any fixed $CK_i = ck_0$, on random variables $a_i, a'_i, t'_i, \omega'_1 \ldots, \omega'_n$, $(d'_{i,1}, d'_{i,2}, \ldots, d'_{i,n}, c_i)$ and $(d''_{i,1}, d''_{i,2}, \ldots, d''_{i,n}, c'_i)$ are computationally indistinguishable.*

Proof. By using the assumption of Variant Decision Diffie-Hellman, $(\omega_1, \omega_2, \ldots, \omega_n, \omega_1^{a_i}, \omega_2^{a_i}, \ldots, \omega_n^{a_i}, g^{a_i})$ and $(\omega_1, \omega_2, \ldots, \omega_n, u_1, u_2, \ldots, u_n, g^{a'_i})$ are computationally indistinguishable where $a_i, a'_i, u_1, u_2, \ldots, u_n \in Z_q$. Let U_i $L(x) = c'_i x + CK'_i mod q$ be a randomly selected line for $c_i = g^{a_i} mod p$, $CK_i = ck_0$ and $k_{ij} = (\omega_j^{a_i} mod p) mod q$, where $1 \leq j \leq n$. Then, $d_{i,j}$ and $d'_{i,j}$ are computed by using equations $d_{i,j} = L(k_{i,j}) mod q$ and $d'_{i,j} = k_{i,j} \oplus dd_{i,j},$. Therefore, $(d'_{i,1}, d'_{i,2}, \ldots, d'_{i,n}, g^{a_i} mod p)$ and $(\overline{d}'_{i,1}, \overline{d}'_{i,2}, \ldots, \overline{d}'_{i,n}, g^{a'_i} mod p)$ are computationally indistinguishable, where $\overline{d}_{i,j} = \overline{L(k_{i,j})} mod q$ for $1 \leq j \leq n$. Since for any $\overline{d}^0_{i,j} \in Z_q$ and $\overline{c}_i \in G_q$.

$$Pr[(\overline{d}_{i,1}, \overline{d}_{i,2}, \ldots, \overline{d}_{i,n}, g^{a'_i} mod p) = (\overline{d}^0_{i,1}, \overline{d}^0_{i,2}, \ldots, \overline{d}^0_{i,n}, g^{a'_i} mod p)] \quad (4.9)$$

$$= Pr[(\overline{d}'_{i,1}, \overline{d}'_{i,2}, \ldots, \overline{d}'_{i,n}, g^{a'_i} mod p) = (\overline{d}^0_{i,1}, \overline{d}^0_{i,2}, \ldots, \overline{d}^0_{i,n}, g^{a'_i} mod p)] \quad (4.10)$$

$$= \frac{1}{q^{n+1}} \quad (4.11)$$

Thus, $(d'_{i,1}, d'_{i,2}, \ldots, d'_{i,n}, c_i)$ and $(d''_{i,1}, d''_{i,2}, \ldots, d''_{i,n}, c'_i)$ are computationally indistinguishable. □

Third attack model for the security of KAP-PBC is the replay attacks. Replay attacks are the attacks that the message(s) of participant(s) are repeated maliciously. For both communication rounds, participant messages $M_{1,i}$ and $M_{2,i}$ contain timestamp T to overcome possible replay attacks. Moreover, the following assumption for timestamps is always hold.

Definition 4.16. *A timestamp value T is an automatically generated and unique value for each broadcast messages of participants.*

4.3.3. Security of Dynamic Group Operations

Dynamic group key agreement protocols provide efficient key update mechanism when the set of participants is updated. However, updating group key may cause security vulnerabilities as shown in [18], which figures out forward confidentiality and backward confidentiality properties. According to these properties, neither a new participant join the group nor an existing participant leaves the group cannot obtain the former group keys or subsequent group keys. In this part of the security analysis, we show that join and leave operations provide backward confidentiality and forward confidentiality, respectively. Furthermore, we benefit from the difficulty of discrete logarithm problem. Since they are the extended version of single join and single leave operations, our proofs are constructed on mass join and mass leave.

Theorem 4.2. *Under the difficulty of computing the discrete logarithm problem, any leave operation does not violate the forward confidentiality.*

Proof. Let $\mathcal{U} = \{U_1, U_2, U_3, \ldots, U_n\}$ be the set of participants before mass leave operation. Assume that $U_i, U_{i+1}, U_{i+2}, \ldots, U_{i+k}$ be leaving participants, where $i + k < n$. Also assume that one of the leaving participants, $U_j \in \{U_i, U_{i+1}, U_{i+2}, \ldots, U_{i+k}\}$, can obtain the update

key, K' after mass leaving is realized. Let

$$K = (g^{t_1t_2+\ldots+t_{i-1}t_i+t_it_{i+1}+\ldots+t_nt_1} mod p) mod q$$

be the key before mass leave operation and let

$$K' = (g^{t_1t_2+\ldots+t_{i-2}t'_{i-1}+t'_{i-1}t_{i+k+1}+\ldots+t_nt_1} mod p) mod q$$

be the updated key. The $g^{t_{i-2}t_{i-1}+t_{i-1}t_i+t_it_{i+1}}$ part of K is replaced by $g^{t_{i-2}t'_{i-1}+t_{i-2}t'_{i-1}}$ after the leave operation. Therefore, it is computationally difficult to compute new key K' by using the old key K by solving the equation $(g^{t_{i-2}t'_{i-1}}\ mod p) mod q$ or $(g^{t'_{i-1}t_{i+k+1}}\ mod p) mod q$ under the difficulty of discrete logarithm problem. □

Theorem 4.3. *Under the difficulty of discrete logarithm problem, mass join operation does not violate the backward confidentiality.*

Proof. Let $\mathcal{U} = \{U_1, U_2, U_3, \ldots, U_n\}$ be the set of participants before the join operation and $U_{n+1}, U_{n+2}, \ldots, U_{n+m}$ be the joining participants. Let

$$K = (g^{t_1t_2+\ldots+t_{n-1}t_n+t_nt_1} mod p) mod q$$

and let

$$K' = (g^{t_1t_2+\ldots+t_{n-1}t'_n+t'_nt_{n+1}+\ldots+t_{n+m}t_1} mod p) mod q$$

be the updated key before and after mass join operation, respectively. The $g^{t_{n-1}t_n+t_nt_1}$ is the updated part of K. Therefore, in order to obtain the former group key K by using the new key K', the joining participant $U_i \in \{U_{n+1}, U_{n+2}, \ldots, U_{n+m}\}$ has to obtain t_n from $(g^{t_nt_1}$ $mod p) mod q$ or $(g^{t_{n-1}t_n}\ mod p) mod q$. Therefore, under the difficulty of the discrete logarithm problem it is computationally infeasible to compute t_n by solving K_n and K_{n-1}. □

As shown in Theorem 1 and 2, KAP-PBC protocol provides both forward confidentiality and backward confidentiality properties. Since basic security properties of group key agreement protocols are provided by KAP-PBC, it is a good candidate for establishing a secure communications among participants for applications like instant messaging and telecommunications.

4.3.4. Partial Backward Confidentiality

Partial backward confidentiality is a new security property for dynamic group key agreement protocols. The related operation is defined in KAP-PBC as $joinPBC(\cdot)$ function. In this analysis, we show that a new participant can compute the last valid group key just before joining the group but the new participant cannot compute former group keys.

Let $\mathcal{U} = \{U_1, U_2, U_3, \ldots, U_n\}$ be the set of participants that use KAP-PBC to generate the group key. Let K_{T-1} and K_T be the group keys, where $K_{T-1} \neq K_T$ at time $T-1$ and T, respectively. Also assume that at time $T+1$, participants $U_{n+1}, U_{n+2}, \ldots, U_{n+m}$ joins the group and after the execution of $joinPBC(\cdot)$ function, then the following definition and the corresponding theorem holds.

Definition 4.17. *When $joinPBC(\cdot)$ is executed for m joining participants, where $m \geq 1$, only the first joining participant can compute the former group key. In other words if participants $U_{n+1}, U_{n+2}, \ldots, U_{n+m}$ are the joining participants, then $\mathcal{U} \cup \{U_{n+1}\}$ executes $joinPBC(\cdot)$ function. At time T, the group key is*

$$K_T = g^{t_1 t_2 + t_2 t_3 + \ldots + t_{n-1} t_n + t_n t_1}$$

After the execution of $joinPBC(\cdot)$, at time $T+1$, U_{n+1} computes K_T by using the sendSecretKey$(\cdot)$ messages of participants in $\mathcal{U}$.

Theorem 4.4. *Any joining participant, who involved into the execution of $joinPBC(\cdot)$ function at time $T+1$, is not able to compute the group key at time $T-1$.*

Proof. Let K_0 be the group key for the set of participants $\mathcal{U} = \{U_1, U_2, \ldots, U_n\}$ at time T_0 is $K_0 = (g^{t_1t_2+t_2t_3+\ldots+t_nt_1} mod p) mod q$. Assume that U_{n+2} be the joining participant at time T_2. Then, either one of the following three cases have occurred:

- Let U_{n+1} be the joining participant at time T_1, $\mathcal{U} \cup U_{n+1}$ executes $joinPBC(\cdot)$ and $join(\cdot)$ functions to compute the following group key:

$$K_1' = (g^{t_1t_2+\ldots+t_{n-1}t_n'+t_n't_{n+1}+t_{n+1}t_1} mod p) mod q$$

- Let U_i be the leaving participant at time T_1, $\mathcal{U} - U_i$ executes $leave(\cdot)$ function (participants except U_{i-1} and U_{i-2} just execute $verifyPublicKey(\cdot)$ and $verifySecretKey(\cdot)$ functions) to compute the following group key:

$$K_1'' = (g^{t_1t_2+\ldots+t_{i-2}t_{i-1}'+t_{i-1}'t_{i+1}+\ldots+t_{n+1}t_1} mod p) mod q$$

- Let U_j be the candidate participant to execute $rekey(\cdot)$ function. At time T_1, the resulting group key after the execution of $rekey(\cdot)$ function is as follows:

$$K_1''' = (g^{t_1t_2+\ldots+t_{j-1}t_j'+t_j't_{i+1}+\ldots+t_{n+1}t_1} mod p) mod q$$

At time T_2, joining participant U_{n+2} can compute either K_1', K_1'' or K_1'''. For computing K_0, U_{n+2} has to obtain one of the following according to the executed function at time T_1

- t_n from $(g^{t_nt_1}\ mod p) mod q$ or $(g^{t_{n-1}t_n}\ mod p) mod q$ by using K_1',
- t_{i-1} from $(g^{t_{i-2}t_{i-1}}\ mod p) mod q$ or $(g^{t_{i-1}t_i}\ mod p) mod q$ and t_i from $(g^{t_{i-1}t_i}\ mod p) mod q$ or $(g^{t_it_{i+1}}\ mod p) mod q$ by using K_1'',
- t_j from $(g^{t_{j-1}t_j}\ mod p) mod q$ or $(g^{t_jt_{j+1}}\ mod p) mod q$ by using K_1''',

Under the difficulty of discrete logarithm problem, it is computationally infeasible for participant U_{n+2} to compute K_0 at time T_2. Hence, by using $joinPBC(\cdot)$ function, a joining participant can only compute the last valid group key just before joining the group that is at

time T_1 but cannot compute group keys that is at time T_0 and before . □

An implication of Theorems 1, 2 and 3 is the fact that when a participant leaves the group and then the same participant joins the group again, group keys will all be different before and after the leave and the join.

Given the backward confidentiality, joining participants are not able to view the previous communications for instant messaging applications. On the other hand, for applications of group key agreement protocols in FSS, all of the communication is realized over the shared file. When a file is shared with the joining participant, all of the file content will be available for the participant. Thus, there is no need for a tight backward confidentiality definition in FSS. Therefore, with partial backward confidentiality, we relaxed the definition of backward confidentiality by allowing a joining participant to compute the last valid group key just before joining the group. Moreover, the provision of backward confidentiality overcome the following problems:

- Lack of Privacy:Since TTPs are not involved in file sharing, the shared file can only be viewed by the group members.
- Violation of Availability: There is no need for dedicated participants such as group managers to grant access permissions for joining participants. There exist group administrators but they have the same computation power with the rest of the participants in the group. Group administrator participants are only responsible for initiating the group communications and updating the group key periodically. Moreover, the group administrators are easily replaced by the other participants in the group.
- Escrow-Freeness: Since all of the participants have the group key, there is no need to use backup mechanism for emergency situations.

As shown in the security analysis, KAP-PBC protocol satisfies both the basic security properties of group key agreement protocols and the security properties of the dynamic group key agreement protocols. In addition, security analysis of KAP-PBC against impersonation, eavesdropping and replay attacks is given. Moreover, we show that the KAP-PBC also has

partial backward confidentiality property. Furthermore, KAP-PBC protocol can overcome the possible problems of using group key agreement protocols in FSS such as lack of privacy, violation of availability and escrow-freeness.

4.4. Performance Analysis of KAP-PBC

In this section, we give the performance analysis of KAP-PBC with respect to communications cost and computational complexity cost. For the communications cost analysis, the total number of communications among participants and the length of the transmitted messages by each participant are key features. For computational complexity cost analysis, the computational effort for computing the group key is analyzed. Details of communications cost analysis and computational cost analysis are given in the following subsections.

4.4.1. Communications Cost Analysis

Communications cost analysis of KAP-PBC is based on the total length of transmitted messages by each participant. The communication cost of KAP-PBC is represented with $\mathcal{C}_T$. Since $sendPublicKey(\cdot)$ and $sendSecretKey(\cdot)$ are the only functions that participants communicate, we first analyze the total length of the transmitted messages in these function. Then, we evaluate the communication cost of the protocol and dynamic operations regarding the number of executions of $sendPublicKey(\cdot)$ and $sendSecretKey(\cdot)$ functions.

Definition 4.18. *Length of the Transmitted Messages. Messages that are broadcasted by each participant $U_i \in \mathcal{U}$ in $sendPublicKey(\cdot)$ and $sendPublicKey(\cdot)$ are $\{\omega_i, e_{1,i}, s_{1,i}, T_{1,i}\}$ and $\{T_{1,i}, e_{2,i}, s_{2,i}, c_i, d'_{i,1}, d'_{i,2}, \ldots, d'_{i,n}\}$, respectively. Since all of operations in the broadcast messages are in modular base, the length of a transmitted message is represented by the cardinality of the bit representation of the modular base. For instance, the length of $|x^y mod\, p| = p$ bits.*

For public-key distribution, each participant transmits,

$$\begin{aligned} |sendPublicKey(\cdot)| &= |\omega_i| + |e_{1,i}| + |s_{1,i}| \\ &= p + q + q \\ &= (2q + p)\ bits. \end{aligned}$$

For secret-key distribution, each participants transmits,

$$\begin{aligned} |sendSecretKey(\cdot)| &= |c_i| + |d'_{i,1} + \cdots + d'_{i,n}| + |e_{2,i}| + |s_{2,i}| \\ &= p + (n-1)q + q + q \\ &= ((n+1)q + p)\ bits \end{aligned}$$

During the execution of KAP-PBC, $sendPublicKey(\cdot)$ and $sendSecretKey(\cdot)$ functions are only executed once by each participant. Thus, the communication cost for the execution of KAP-PBC for each participant is:

$$\mathcal{C}_{\mathcal{T}} = ((n+3)q + 2p) \text{ bits}$$

Definition 4.19. *Total Communication Cost. The functions of KAP-PBC protocol that participants communicate with other participants in the group are $sendPublicKey(\cdot)$ and $sendSecretKey(\cdot)$. Therefore, for set of participants, $\mathcal{U} = \{U_1, U_2, \ldots, U_n\}$, communication cost, $\mathcal{C}_T$, in different execution cases of KAP-PBC are as shown as below:*

(i) For Default Execution of KAP-PBC: For group with n participants, the total communication cost of executing KAP-PBC is:

$$\mathcal{C}_{\mathcal{T}} = \frac{n(n-1)}{2}((n+3)q + 2p) \text{ bits}$$

(ii) Join operation: When a new participant joins the group, the group key must be updated. Key update is realized by either single join or mass join. For all join variants, both all of

the participants in $\mathcal{U}$ have to broadcast the resulting messages of $sendPublicKey(\cdot)$ and $sendSecretKey(\cdot)$ functions. Therefore, for k joining participants and n participants in the group, the total communication cost for join operations is

$$\mathcal{C}_T = 2(n+k-1)((n+3)q+2p) \text{ bits}$$

(iii) For Leave Operation: KAP-PBC protocol also provides both single and mass leave operations by using $leave(\cdot)$ function. Let $U_i \in \mathcal{U}$ be the leaving participant. Then, the total number of communications in $leave(\cdot)$ function is only three, because, $sendPublic$-$Key(\cdot)$ function is only executed by U_{i-1} and $sendSecretKey(\cdot)$ function is executed by U_{i-1} and U_{i-2}. Therefore, for group with n participants, the total communication cost is

$$\mathcal{C}_T = 2(n+2)q+3p \text{ bits}$$

(iv) For Key Update Operation: The total number of communications in this operations is also the same as in $leave\cdot$ function. For $rekey(\cdot)$ function, there is no new participants in the group. Hence, only the selected participants execute $sendPublicKey(\cdot)$ and $sendSecretKey(\cdot)$ functions. When a random participant U_i is selected to update the key, $sendPublic$- $Key(\cdot)$ and $sendSecretKey(\cdot)$ functions are executed by U_i and $sendSecretKey(\cdot)$ function is executed by U_{i-1}. Therefore, the communication cost for n participants is

$$\mathcal{C}_T = 2(n+2)q+3p \text{ bits}$$

(v) For Join Operation with Partial Backward Confidentiality: This operation is realized with $joinPBC(\cdot)$ function. According to the definition of this function, whether it is mass join or a single join, only one joining participant can compute the last group key before joining the group. Therefore, for n participants in the group, the total communication cost for join operation with partial backward confidentiality is equal to the

execution of join operation for one new participant, which is

$$\mathcal{C}_T = (2n+1)((n+3)q+2p) \text{ bits}$$

4.4.2. Computational Complexity Cost Analysis

In this section, we analyze the computation cost of KAP-PBC with respect to modular exponentiation operations. During the computation of the group key, each participant makes multiplications, summations, logical XORs and exponentiation in modular base. The most time-consuming operations among these operations is the modular exponentiation. Therefore, we only concentrate on the computation complexity analysis of KAP-PBC protocol with respect to modular exponentiations, denoted as T_{exp} and the computation cost is denoted as $\mathcal{C}_C$. Although exact form of modular exponentiation is different for different functions, computation complexity of modular exponentiation could be defined as $T_{exp} = O(x^y mod z)$.

Let $\mathcal{U} = \{U_1, U_2, U_3, \ldots, U_n\}$ be the set of participants, the computational complexity cost analysis is as follows:

(i) For $sendPublicKey(\cdot)$ function,

$$\mathcal{C}_C(sendPublicKey) = 2T_{exp} = O(1)T_{exp}$$

(ii) For $verifyPublicKey(\cdot)$ function,

$$\mathcal{C}_C(verifyPublicKey) = 2nT_{exp} = O(n)T_{exp}$$

(iii) For $sendSecretKey(\cdot)$ function,

$$\mathcal{C}_C(sendSecretKey) = (n+3)T_{exp} = O(n)T_{exp}$$

(iv) Finally, for $verifySecretKey(\cdot)$ function,

$$\mathcal{C}_C(verifySecretKey) = (3n)T_{exp} = O(n)T_{exp}$$

Therefore, for each participant, the total computation cost of executing KAP-PBC is.

$$\mathcal{C}_C = (5n+5)T_{exp} = O(n)T_{exp}$$

Given the unit computation costs of the functions of KAP-PBC, we analyze the computation cost of dynamic group operations with respect to the number of function executions as follows:

(i) For Join, $join(\cdot)$: Let $U_{n+1}, U_{n+2}, \ldots, U_m$ be the joining participants for $\mathcal{U}$. Then, for adding m participants into the group, $m+1$ participants executes $sendPublicKey(\cdot)$ function and $m+2$ participants executes $sendSecretKey(\cdot)$ function. For the verification processes, $verifyPublicKey(\cdot)$ is executed by $m+1$ participants and $verifySecretKey(\cdot)$ function is executed by $m+2$ participants. Therefore, the computation cost of join operation is $\mathcal{C}_C = O(m+n)T_{exp}$ for each joining participant and $\mathcal{C}_C = O(m+n)T_{exp}$ for the participant U_n and U_{n+1}. Since the rest of the participants only verifies the broadcast messages, for participants $U_i \in \{U_1, U_2, \ldots, U_{n-2}\}$, the computation cost is $\mathcal{C}_C = O(m)T_{exp}$.

(ii) For Key Update, $re-key(\cdot)$: Let U_i be the randomly selected candidate participant for updating the group key. Then, only U_i executes the $send\text{-}\ PublicKey(\cdot)$ function and U_i and U_{i-1} execute $verifyPublicKey(\cdot)$ function for only the broadcast messages of U_i. The $sendSecretKey(\cdot)$ function and the corresponding $verifySecretKey(\cdot)$ function are executed only for U_{i-1} and U_i. Therefore, the computation cost cost of executing $rekey(\cdot)$ function is $\mathcal{C}_C = O(1)T_{exp}$ for participants that only execute $verifyPublicKey(\cdot)$ and $verifySecretKey(\cdot)$ functions and $\mathcal{C}_C = O(n)T_{exp}$ for U_i and U_{i-1}.

(iii) For Join with PBC, $joinPBC(\cdot)$: Since the join with backward confidentiality operation is independent of the number of joining participants, this operation has the

same computation cost with key update, which is $\mathcal{C}_C = O(1)T_{exp}$ for the execution of $verifyPublicKey(\cdot)$ function and $verifySecretKey(\cdot)$ function. Moreover, $\mathcal{C}_C = O(n)T_{exp}$ for U_i and U_{i-1}.

(iv) For Leave, $leave(\cdot)$: Let U_i be the leaving participant, where $\mathcal{U}' \subset \mathcal{U}$. After participants leave the group, the group key is updated by using the similar assumption with key update operation. U_{i-1} executes $sendPublicKey(\cdot)$ function and the other participants in the group executes $verifyPublicKey(\cdot)$ for only the broadcast messages of U_i. Later, the $sendSecretKey(\cdot)$ function is only executed by U_{i-1} and U_{i-2}. Then, participants in the group execute $verifySecretKey(\cdot)$ function based on the broadcast messages of U_{i-1} and U_{i-2}, which is also equal to the computation cost of the key update function, $\mathcal{C}_C = O(1)T_{exp}$ for participants that only execute $verifyPublicKey(\cdot)$ and $verifySecretKey(\cdot)$ functions. On the other hand, for U_i and U_{i-1}, the computation cost of leave function is $\mathcal{C}_C = O(n)T_{exp}$.

As shown in the communication cost analysis and computation cost analysis, the length of the transmitted messages and the number of modular exponentiations calculated during the execution of functions as $verifyPublicKey(\cdot)$, $sendSecretKey(\cdot)$ and $verifySecretKey(\cdot)$ depend on the number of participants in the group. Therefore, when the number of participants increases, communication and computation costs of executing KAP-PBC increases. On the other hand, we see the existence of dynamic group operations decreases the costs of communications and computations in terms of updating the group key.

4.4.3. Performance Comparisons for Group Key Agreement Protocols

We compare the performance of KAP-PBC with protocols DASGKA in [110] and DCKAP in [16] since they provide almost all of the criteria for the comparisons in Table 4.1. In Table 4.2, we compare group key computation of DASGKA, DCKAP and KAP-PBC protocols. The total computational complexity cost of all protocols are $O(n^2)T_{EXP}$ for the set of participants $\mathcal{U} = \{U_1, U_2, \ldots, U_n\}$. Therefore, each participant has $O(n)T_{EXP}$ cost for the group key computation. For the communications cost, DASGKA has better performance than both DCKAP and KAP-PBC.

Table 4.2. Comparisons of the Total Communications Cost and Total Computational Complexity Cost

Protocols	**Computational Complexity Cost**	**Communications Cost**
DASGKA	$O(n^2)T_{EXP}$	$O(1)\|p\|$ bits
DCKAP	$O(n^2)T_{EXP}$	$O(n^2)\|q\| + O(n)\|p\|$ bits
KAP-PBC	$O(n^2)T_{EXP}$	$O(n^2)\|q\| + O(n)\|p\|$ bits

In Table 3, we compare DASGKA, DCKAP and KAP-PBC with respect to the total computational complexity costs of join and leave operations. Let $\mathcal{U} = \{U_1, U_2, \ldots, U_n\}$ be the set of participants before join and leave operations. For joining participants $\{U_{n+1}, U_{n+2}, \ldots, U_{n+m}\}$ and leaving participants $\{U_{i+1}, U_{i+2}, \ldots, U_{i+m}\}$, all protocols provides the same computational complexity cost for joining and leaving operations.

Table 4.3. Comparison of the Total Computational Complexity Costs for Join and Leave Operations

Protocols	**Computational Complexity Cost**	
	Join	**Leave**
DASGKA	$O(mn)T_{EXP}$	$O(n)T_{EXP}$
DCKAP	$O(mn)T_{EXP}$	$O(n)T_{EXP}$
KAP-PBC	$O(mn)T_{EXP}$	$O(n)T_{EXP}$

Since DASGKA uses different mathematical operations for group key computation, this protocol produce shorter broadcast messages with respect to DCKAP and KAP-PBC. For a real-life implementation, the group key computation time and the group key update time proportionally increases with the number of participants in the group. In Section 7.6, we have presented our comments for the implementation issues of KAP-PBC and PFSS.

4.5. PFSS: Private File Sharing System, An Application of KAP-PBC

In this section, we define a secure file sharing system, namely Private File Sharing System (PFSS). The proposed system is an application for the use of group key agreement protocol on file sharing systems to solve lack of privacy, violation of availability and dependency for key-escrow problems.

4.5.1. PFSS Overall View

The overall organization PFSS components is shown in the Figure 4.15 and PFSS components can be defined as follows:

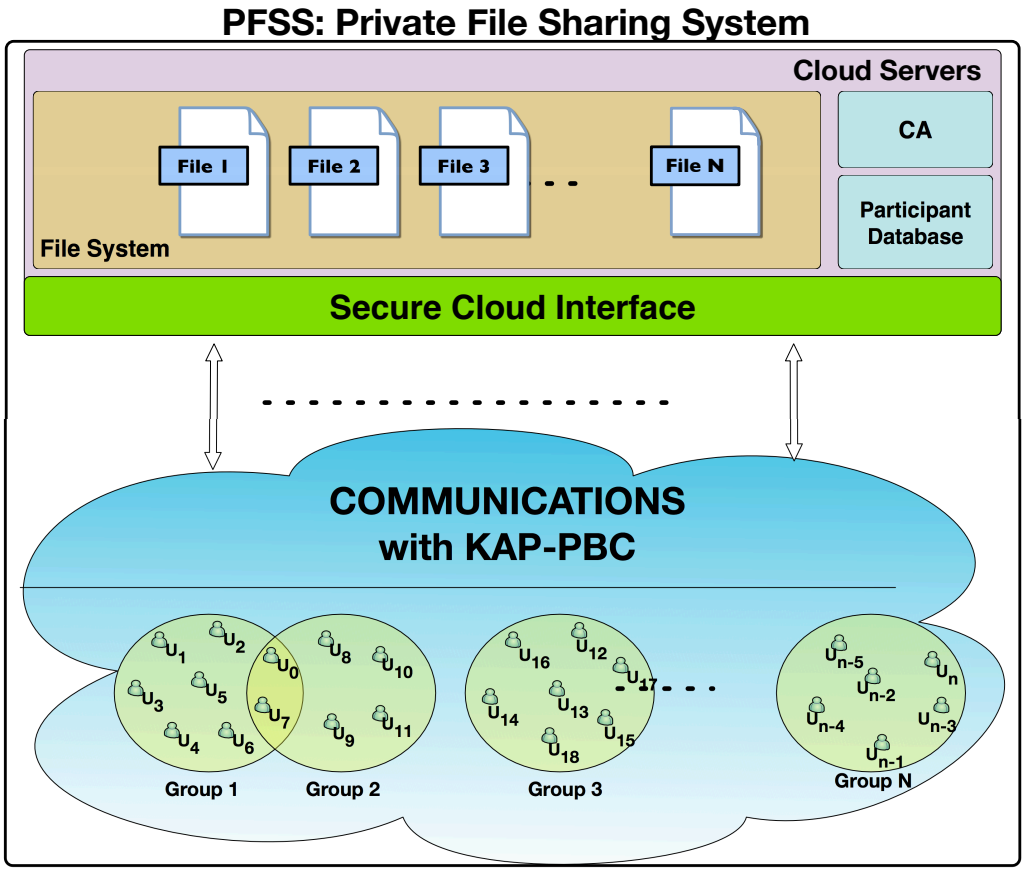

Figure 4.15. PFSS Overall View

(i) Participant Groups: Let $\mathcal{U}_1, \mathcal{U}_2, \mathcal{U}_3, \ldots \mathcal{U}_N$ be the groups of PFSS. Each group U_i, where $1 \leq i \leq N$ has its own set of participants, $\mathcal{U}_i = \{U_1, U_2, U_3, \ldots, U_{|\mathcal{U}_i|}\}$.

(ii) KAP-PBC Protocol: In PFSS, each group $\mathcal{U}_i$ uses group key to store the shared files in an encrypted format. In PFSS, such group keys are generated by using KAP-PBC.

(iii) Secure Cloud Interface: PFSS provides cloud interface. It is required that cloud interface be secure for example by using TLS 1.2 [122] since participant identities and passwords will be defined through this interface.

(iv) Participant Database: PFSS provides registration system for new users. After the registration is completed, users become the possible participants of file sharing groups in PFSS. Login credentials and information of groups that a participant involved are stored in participant database in PFSS.

(v) Certificate Authority: After the registration of a new user is completed, each participant U_i has to generate and certify long-term key pairs in order to join file sharing. Then, the public key of each participant U_i is certified by using CA in PFSS. In this thesis, we assume that CA is TTP. In case of malicious CA, we can assume that participants will be able to employ an decision algorithm to decide on the security of certificates from different CAs, as in the Byzantine Generals Problem.

(vi) File System Files: PFSS has a file system in its cloud servers to organize the files as shared files. A shared file could as well be a directory identifying a tree of shared files.

4.5.2. PFSS Services

Using dynamic group key agreement protocols as a confidentiality service is a challenging issue in file sharing systems because of the backward confidentiality property blocks a joining participant computing the last valid group key just before joining the group. TTPs or dedicated participants are used to overcome this issue. However, if TTPs are involved in file sharing, the privacy of the file is violated. If dedicated participants exist, there is a possibility for single-point-of-failure. Also, the necessity for a key escrow mechanism arises to provide recovery mechanism. In this chapter, we have proposed the partial backward confidentiality property to disregard the need of TTPs and dedicated participants by enabling one of the joining participants to compute the last valid group key just before joining the group. PFSS is an example application of KAP-PBC to show the applicability of partial backward confidentiality.

Definition 4.20. *General Assumptions for PFSS. We assume that the following definitions, properties, services and mechanisms are exist for PFSS:*

- *The communications between PFSS and participants are realized in a channel secured with TLS.*
- *A revocation list is used to store certificates of malicious participants. Let U_i be the participant in file sharing group. The certificate of U_i is added into revocation list in the following cases:*
  - *U_i tries to disrupt the group key computation,*
  - *U_i tries to gain unauthorized access for shared files,*
  - *U_i refuses to renew the certificate of its public-key after the certificate has expired, or*
  - *U_i tries to replace the shared file by some file.*
- *A Majority Rule is used as a decision mechanism in groups such as adding new participants or deciding the group termination. In case of equality, the vote of the group administrator is taken into account.*
- *All encryption and decryption operations are realized in the computers of participants by using the group key.*
- *Access permissions of update allowed files are set by file administrator in the group formation. If new participants join the file sharing, then access permissions for new participants are set by using the majority rule.*
- *Since files are stored in the cloud server, if a participant U_i of a file-sharing group wants to make changes on a shared file, first, the file has to be downloaded to the computer of participant, decrypted by using the group key and then the changes can be applied on plaintext. Finally, the modified file is uploaded on the cloud server of PFSS.*
- *If all participants of a group leave, then the group is terminated. and the files associated with the group is deleted.*

Definition 4.21. *User Registration and Certification Service. To be able to use PFSS, a user has to register with PFSS. Let U_i be the new participant that wants to register with PFSS. First, user registers by using the cloud interface by entering the necessary information such*

as username, password and e-mail address. If user is accepted by PFSS, then an approval notice is sent to U_i and user information is stored in the participant database. After the registration, each participant U_i has to generate its long-term key pair for public key certification. Unless the public-key of a participant U_i is certified, PFSS does not allow U_i to use file sharing. For certification of public-key, U_i sends (ID_i, y_i) to the PKCA and PKCA responds with $Certified(ID_i, y_i)$ to U_i.

Definition 4.22. *Group Formation Service. In PFSS, group formation is started with an invitation. A participant creates a new file and invites other participants to share the file is called the File-Administrator. Let U_i be the file-administrator. Then, group generation process is realized as shown in Figure 4.16.*

1. U_i selects set of participants $\mathcal{U} = \{U_1, U_2, \ldots, U_n\}$ for sharing file from participant database.
2. U_i sends invitation to the participants in $\mathcal{U}$ by using the $invite(\cdot)$ function of KAP-PBC.
3. Let $\mathcal{U}_N$ be the participants who replied with negative answer or with no answer within a specific timeout period. Then, U_i re-organizes $\mathcal{U}$ as $\mathcal{U} = \mathcal{U} - \mathcal{U}_N$.
4. The information of the group, $\mathcal{U}$, and the access permissions are recorded in participant database by file-administrator.
5. Each participant in $U_i \in \mathcal{U}$ group key for files to be shared.

Figure 4.16. Group Formation Service

Definition 4.23. *File Confidentiality Service. Before uploading the file into the file system of PFSS, the shared file must be encrypted by using the group key. In PFSS, the encryption and decryption algorithm, algorithm mode and IV if needed are recorded as part of file metadata in File System. In this secure file sharing system, AES-256 in CBC mode has been used as encryption/decryption algorithm.*

Definition 4.24. *Group Key Update Service. One of the critical issues for file sharing systems is that the handling dynamic groups. KAP-PBC provides dynamic group operations such as join, leave and key update. Let $\mathcal{U} = \{U_1, U_2, \ldots, U_n\}$ be the set of participants in the file sharing group and K_t be the group key at time t. The details for the use of KAP-PBC dynamic group operations in PFSS are as follows:*

(i) *Join Operation: PFSS uses* $join(\cdot)$ *and* $joinPBC(\cdot)$ *functions for adding new participants into the group. Let* $U_{n+1}, U_{n+2}, \ldots, U_{m+n}$ *be the joining pariticipants for the file sharing group* $\mathcal{U}$ *at time* t*. Then, the following functions are executed for the given participant sets to compute the updated keys:*

$$K'_{t+1} = \mathcal{U} \cup U_{n+1} \rightarrow joinPBC(\cdot)$$

$$K''_{t+1} = \mathcal{U} \cup U_{n+1} \cup U_{n+2} \cup \cdots \cup U_{n+m} \rightarrow join(\cdot)$$

Since U_{n+1} *is in the execution of* $joinPBC(\cdot)$ *function,* U_{n+1} *can compute* K_t *by using* K'_{t+1}*. Then,* U_i *downloads and decrypts the shared files by using* K_t*. Then, encrypts the shared files with* K''_{t+1} *and uploads onto the cloud server.*

(ii) *Leave Operation: Let* U_i *be the leaving participant from the file sharing group* $\mathcal{U}$ *at time* t*. Then, the following function is executed for the given participant sets to compute the updated key:*

$$K'_{t+1} = \mathcal{U} - U_i \rightarrow leave(\cdot)$$

Then, participant U_{i-1} *updates the file by using* K'_{t+1}*. While computing new key* $K'_{t+1} = \mathcal{U} - U_i \rightarrow leave(\cdot)$*, participants except* U_{i-1} *and* U_{i-2} *just execute* $verify-PublicKey(\cdot)$ *and* $verifySecretKey(\cdot)$ *functions.*

(iii) *Key Update Operation: In order to increase security of shared files PFSS uses the key update function,* $rekey(\cdot)$*, of KAP-PBC to update the key. A participant* U_i *is randomly selected from the group* $\mathcal{U}$ *at time* t*. Then, the following function is executed for the given participant sets to compute the updated key:*

$$K'_{t+1} = \mathcal{U} \rightarrow rekey(\cdot)$$

After the execution of $rekey(\cdot)$ *is completed.* U_i *downloads and decrypts the shared files from PFSS cloud server by using* K_t*. Then,* U_i *encrypts the shared files with the*

new key K'_{t+1}. While computing new key by using rekey$(\cdot)$ function, participants except U_{i-1} and U_i just execute verify− PublicKey$(\cdot)$ and verifySecretKey$(\cdot)$ functions.

4.5.3. PFSS File Sharing Scenarios

In this section, we give the example scenarios for PFSS. Let U_1 be the participant that wants to share a file, F, with other participants in PFSS, then the file sharing is realized in the following order:

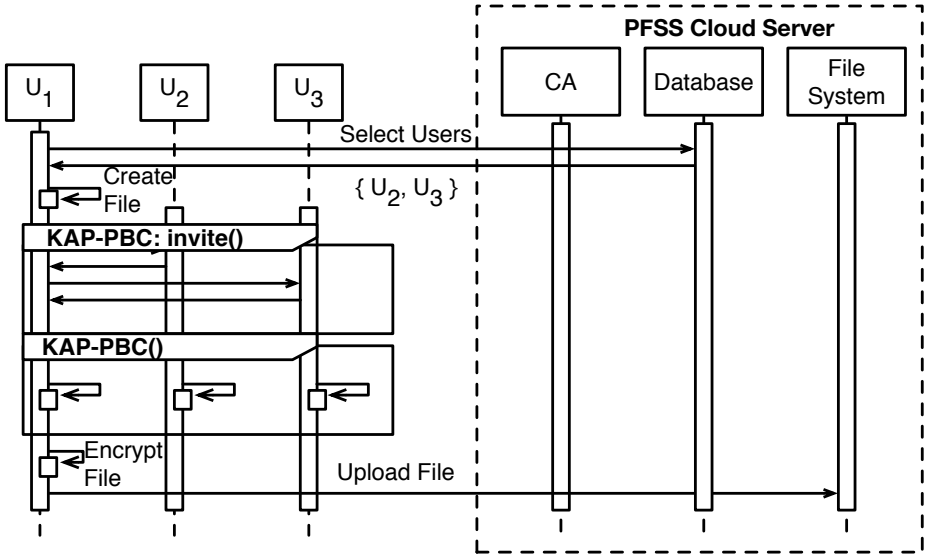

Figure 4.17. UML Sequence Diagram for PFSS Create Group and File Scenario

Definition 4.25. *Scenario 1: Create Group and File. Let U_1 be the participant that wants to share the file F with other participants U_2 and U_3 as shown in Figure 3. Then, create group and file scenario operates as follows:*

(i) U_1 selects participants U_2 and U_3 and retrieves certificate information from PFSS database.

(ii) U_1 creates a F for sharing.

(iii) U_1 sends invitation to participants U_2 and U_3 by using invite$(\cdot)$ function.

(iv) Since it is the group administrator, U_1 forms the group $\mathcal{U} = \{U_1, U_2, U_3\}$.

(v) All of the participants in $\mathcal{U}$ executes KAP-PBC protocol for computing the group key K_1

(vi) U_1 *encrypts* F *to be shared by using the group key and uploads* F *to the PFSS File System.*

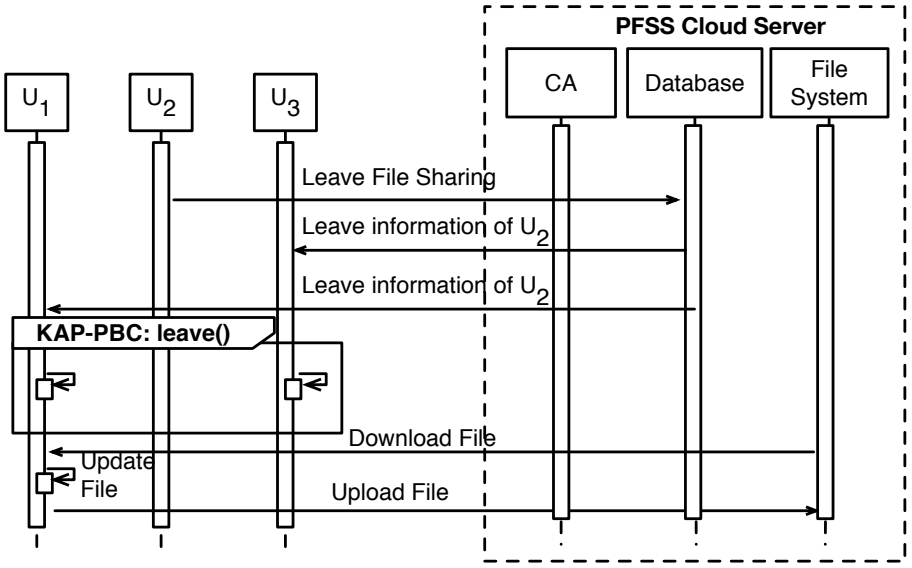

Figure 4.18. UML Sequence Diagram for PFSS Participant Leave

Definition 4.26. *Scenario 2: Participant Leave. Let* $\mathcal{U} = \{U_1, U_2, U_3\}$ *be the set of participants for sharing the file* F *as shown in Figure 4. Then, leaving of participant* U_2 *is given below:*

(i) U_2 *leaves the file sharing and informs* U_1 *and* U_2 *by using PFSS.*

(ii) After "I am leaving file sharing" information is send from PFSS database, U_1 *and* U_3 *executes* $leave(\cdot)$ *to update the group key as* K_2*.*

(iii) U_1 *updates* F*.*

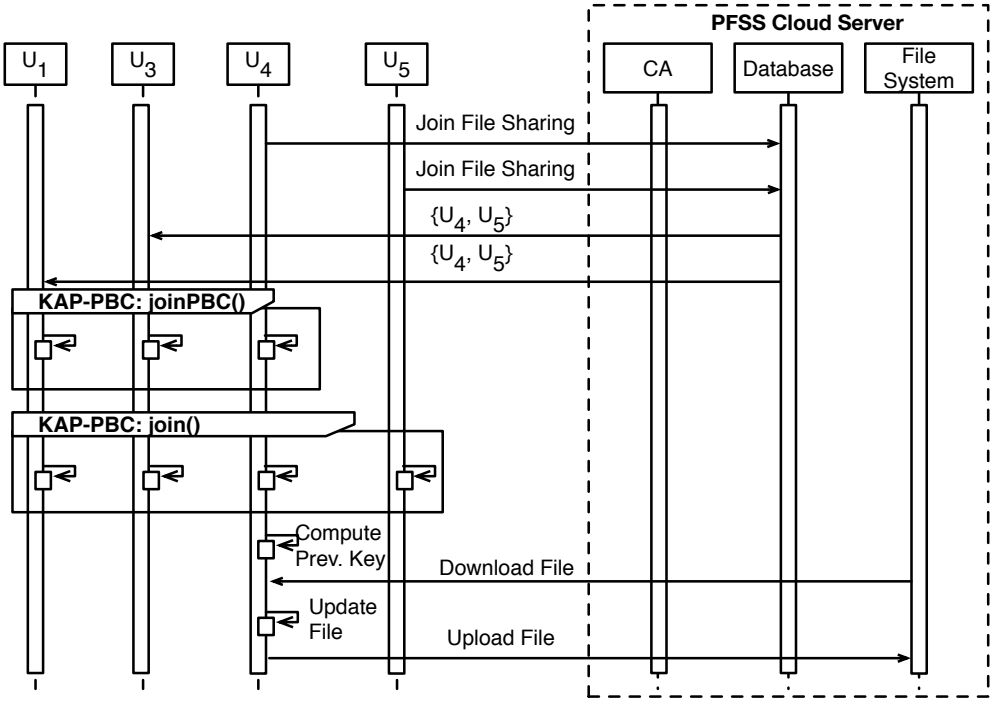

Figure 4.19. UML Sequence Diagram for PFSS Participant Join

Definition 4.27. *Scenario 3: Participant Join. Let $\mathcal{U} = \{U_1, U_3\}$ be the set of participants for sharing the file F. Let U_4 and U_5 be the participants joins the file sharing as shown in Figure 5. Then, the participant join is realized as follows:*

(i) New participant U_5 registers the PFSS.

(ii) U_4 and U_5 joins the group $\mathcal{U}$

(iii) U_1, U_3, U_4 executes $joinPBC(\cdot)$ function to compute group key K_3.

(iv) U_1, U_3, U_4, U_5 executes $join(\cdot)$ function to compute group key K_4.

(v) U_4 updates F.

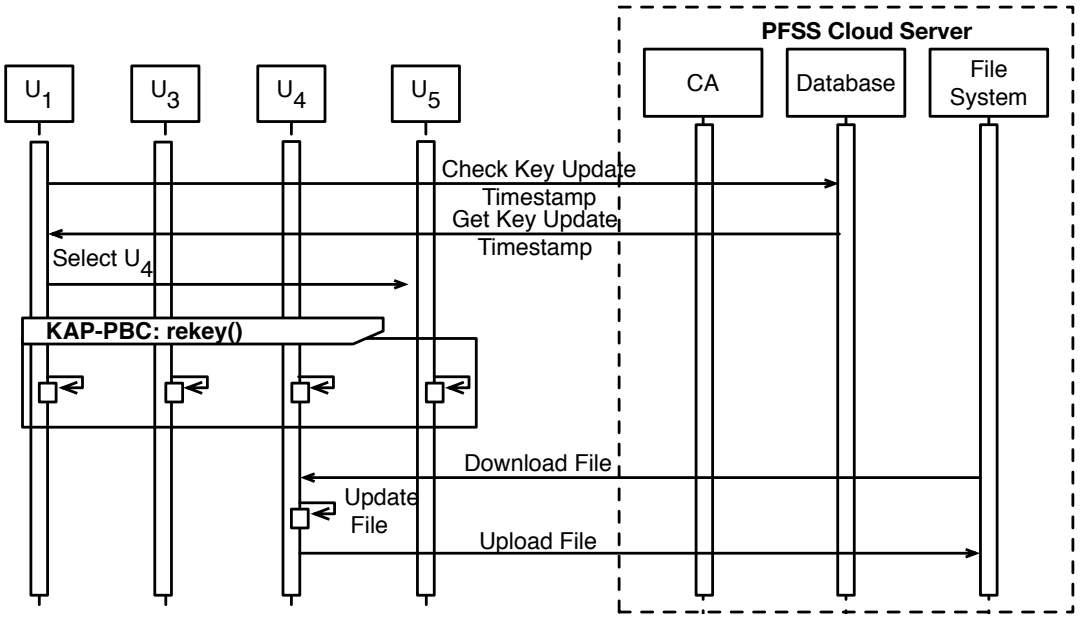

Figure 4.20. UML Sequence Diagram for PFSS Key Update

Definition 4.28. *Scenario 4: Group Key Update. Let $\mathcal{U} = \{U_1, U_3, U_4, U_5\}$ be the set of participants for sharing the file F. The key update operation as shown in Figure 4.20 is realized as follows:*

(i) The file administrator, U_i, checks PFSS database for the key-update-period.

(ii) If the key-update-period exceeds, then U_1 randomly selects a participant U_4 in the group to update the group key.

(iii) U_1, U_3, U_4, U_5 executes rekey$(\cdot)$ function to compute group key K_5.

(iv) U_4 updates F.

4.5.4. Simulations of PFSS Group Formation and Group Update Services

Currently, our view of PFSS does not involve very large files in the order of tera/peta bytes since our goal is to demonstrate the applicability of KAP-PBC. We assume that the executions of file confidentiality and registration services are standard operations. Therefore, we only consider the simulations of KAP-PBC execution in group formation and group key update services. Simulations were carried out by using Python 2.7 and Charm Crypto Framework [103] that run on a MacBook Air 2012 early release with 250 GB SSD disk, 1.8GHz Intel Core i5 processor and 4GB 1600 MHz DDR3 RAM.

For the set of participants $\mathcal{U} = \{U_1, U_2, U_3, \ldots, U_{20}\}$, the simulations of KAP-PBC and dynamic group operations are as shown in Figure 4.21. Details of the simulations are as follows:

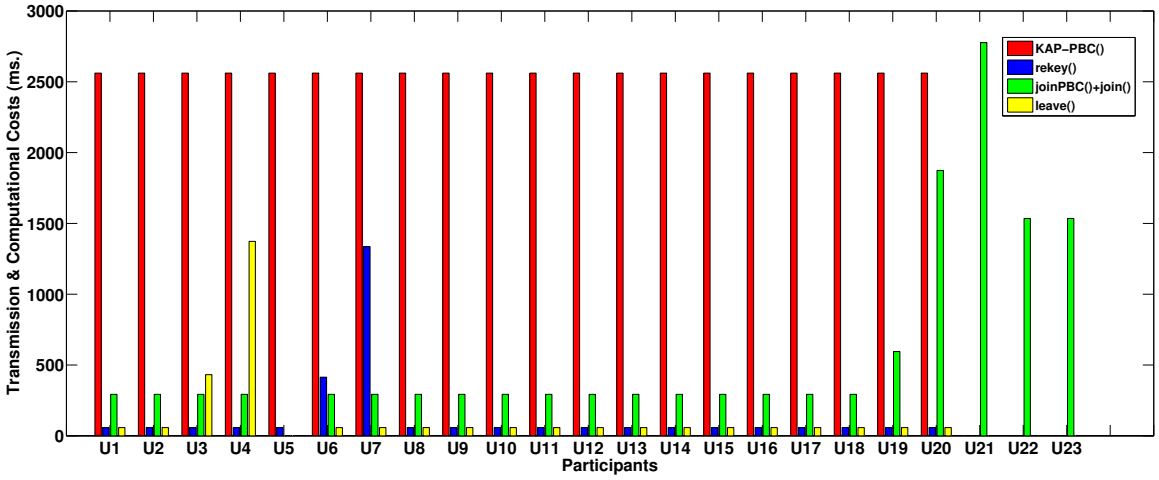

Figure 4.21. Simulations for KAP-PBC Functions

- PFSS KAP-PBC: Each participant $U_i \in \mathcal{U}$ executes KAP-PBC protocol to compute the group key. For $|\mathcal{U}| = 20$, computation of group key takes 2561 ms for each participant in the group.
- PFSS Key Update: When the key-update-period is exceeded, group key must be updated by using $rekey(\cdot)$ function. As shown in the Figure 7, we assume that U_7 is the randomly selected participant to update the group key. Therefore, total communications and computations costs of participants are $\mathcal{C}_C(U_7) + \mathcal{C}_T(U_7) = 1336ms$, $\mathcal{C}_C(U_6) + \mathcal{C}_T(U_6) = 414ms$ and $\mathcal{C}_C(U_i) + \mathcal{C}_T(U_i) = 58ms$, where $U_i \in \mathcal{U} - \{U_6, U_7\}$.
- PFSS Leave: For participant leave, we assume that participant U_5 leaves file sharing. Therefore, the communication and computation costs of participants are $\mathcal{C}_C(U_4) + \mathcal{C}_T(U_4) = 1373ms$, $\mathcal{C}_C(U_3) + \mathcal{C}_T(U_3) = 432ms$, $\mathcal{C}_C(U_5) + \mathcal{C}_T(U_5) = 0ms$ and $\mathcal{C}_C(U_i) + \mathcal{C}_T(U_i) = 58ms$, where $U_i \in \mathcal{U} - \{U_3, U_4, U_5\}$.
- PFSS Join: For participant join, we assume that participants U_{21}, U_{22}, U_{23} joins the $\mathcal{U}$. According to the definition of participant join in group update service, communications and computational costs of participants are as follows: $\mathcal{C}_C(U_{21}) + \mathcal{C}_T(U_{21}) = 2777ms$, $\mathcal{C}_C(U_{20}) + \mathcal{C}_T(U_{20}) = 1874ms$, $\mathcal{C}_C(U_{22}) + \mathcal{C}_T(U_{22}) = \mathcal{C}_C(U_{23}) + \mathcal{C}_T(U_{23}) = 1535ms$, $\mathcal{C}_C(U_{19}) + \mathcal{C}_T(U_{19}) = 595ms$ and $\mathcal{C}_C(U_i) + \mathcal{C}_T(U_i) = 293ms$, where $U_i \in \{U_1, U_2, \ldots, U_{18}\} - \{U_5\}$.

As expected, the total execution cost of group key update services provides better performance than group formation service. On the other hand, average execution time of KAP-PBC for each participant is approximately two seconds, which is reasonable for a group with twenty participants. On the other hand, as given in the the complexity analysis, execution time of KAP-PBC and dynamic group operations depend on the number of participants in the group. Furthermore, number of participants in the group affects the scalability of KAP-PBC. For instance, communication and computational costs of KAP-PBC are $\mathcal{C}_C(U_i) + \mathcal{C}_T(U_i) \approx 15sec.$ for $U_i \in \mathcal{U}$, where $|\mathcal{U}| = 100$ and $\mathcal{C}_C(U_i) + \mathcal{C}_T(U_i) \approx 17min.$ for $U_i \in \mathcal{U}$, where $|\mathcal{U}| = 1000$. Therefore, KAP-PBC is applicable for small groups up to hundred participants or KAP-PBC is applicable for larger groups if there is no time constraint for compute the group key.

Definition 4.29. *Assumptions for the Implementation of PFSS. Time between join and leave operations is sufficiently long so that new group key can be computed in due time. The related performance study is left as our future work.*

4.5.5. An Overview of File Sharing Systems

PFSS proposes a different file sharing approach by using KAP-PBC to provide solutions for lack privacy, violation of availability and dependency for key escrow problems of file sharing systems. Therefore, we compare PFSS and other protocols in literature as shown in Table 4.4 according to the possessions of the following criteria:

- High Availability: In this thesis, we consider this concept as avoidance of single point of failure to prevent from violation of availability.
- Escrow-Freeness: No authorized third party may gain access to encryption keys of shared file groups even in emergency circumstances.
- File Privacy: Shared files can only be viewed / modified by the group members.

Table 4.4. Comparison of File Sharing Systems

File Sharing Systems	**High Avail.**	**Escrow-Freeness**	**File Privacy**
System in [67]	✗	✓	✓
pNFS-AKE III in [71]	✓	✓	✗
HABE in [79]	✗	✗	✗
Mona in [68]	✓	✗	✓
System in [74]	✓	✓	✗
FADE in [70]	✓	✗	✓
TimePRE in [75]	✗	✗	✗
TT-SFSS	✗	✓	✓
PFSS	✓	✓	✓

According to the Table 4.4, PFSS is the only file sharing system that provides high availability, escrow-freeness and file privacy. In addition to PFSS, FADE, pNFS-AKE III, System in [74] and Mona provide high availability and escrow freeness criteria. Because of the servers have access permission to shared files in the cloud, file privacy is not provided for pNFS-AKE III. For FADE, escrow-freeness criterion is not provided. In HABE and TimePRE, none of the criteria given above are satisfied since access control mechanisms are the major concern for these studies. Finally, System in [67] and TT-SFSS provide escrow-freeness and file privacy properties. The performance analysis of file sharing systems is out of scope of this chapter.

As a consequence, regarding the comparisons with other file sharing systems, PFSS provides better results. Therefore, shared files are securely stored in cloud server and there is no feasible way for cloud server to view the content of the file. Another important property is that the high availability. The re-encryption process is always realized by different users in file sharing group. Therefore, this also makes PFSS less vulnerable than dedicating a specific user (GAs) to re-encrypt the shared file. Finally, since there is no TTP or dedicated participant in PFSS, escrow-freeness property is also satisfied.

4.5.6. Implementation Issues of PFSS

First, we address the important issues with respect to the execution of a file confidentiality service and file upload/download operations. Then, we extend findings in simulation results for the implementation of PFSS. Important issues that may affect the implementation of a file confidentiality service and file upload/download operations are as follows:

- File Confidentiality: As defined in the File Confidentiality Service, PFSS uses AES-256 with CBC mode for an encryption and a decryption of shared files. We have simulated the encryption/decryption of AES-256 with CBC mode by using the same simulation environment. Execution of a file encryption / decryption approximately takes 2622 milliseconds for 10MB file, 3772 milliseconds for 100MB file, 8756 milliseconds for 500 MB file and 14259 milliseconds for 1GB file. Therefore, it is obvious that the execution time of a file confidentiality service increases linearly with respect to the size of a file to be encrypted/decrypted. Moreover, if a more powerful hardware configuration has been used for simulations, we can have a better execution time for the file confidentiality.
- File Upload/Download: File upload and download speed depends on the Internet speed of a participant. Therefore, if participants have high bandwidths for communications, they can easily share large files in the order of giga bytes.

As given in Section 7.4, PFSS provides a reasonable group key execution time for a group of 100 participants, which is approximately 17 seconds. However, as shown in the simulation results, the number of participants in the group affects the group key execution time. For instance, if there exist 200 participants in the group, the group key execution time takes approximately 48 seconds per participant. In Section 7.5, we have compared PFSS with other secure file sharing systems. As shown in the Table 4, file sharing systems [67, 75, 79] use dedicated participants such as group managers for distributing file confidentiality keys and file sharing systems in [71, 74] use TTPs for distributing file confidentiality keys. In general, key distribution systems do not perform as many mathematical operations as in key agreement protocols. For instance, commercial file sharing systems such as Google Drive

and Dropbox can operate on large groups that may have thousands of participants. Therefore, for a real life application, we assume that PFSS is not suitable for groups larger than 100 participants.

Final important comment on the implementation of PFSS is dynamic group operations of KAP-PBC. The proposed protocol provides join, leave, join with the partial backward confidentiality and rekey operations for updating a group key. As shown in Figure 7, an execution time for group key update operations is less than an execution time for computing a group key from the beginning and the execution time of group key update not so dependent as the group key computation. As given in Section 7.4, some of the participants in the group are selected for executing the KAP-PBC from the beginning. For instance, in a set of 100 participants, if five participants join the group, joining participants and the last participant of a group update the group key from the beginning. The rest of the participants only verify new secret keys of joining participants and the last participant in the group. Therefore, in practice, execution time is approximately equal to the execution of KAP-PBC for group of six participants. For leave operation, in a circular list, only participants located before leaving participants execute KAP-PBC from the beginning. For instance, in a set of 100 participants, if five participants, who are located in a consecutive manner in the group, leave the group, only one participant executes the protocol from the beginning. On the other hand, for file sharing systems [67, 75, 79], group key update for join and leave operations is only realized by group managers. Since only group manager updates the file confidentiality key, we assume that file sharing systems in [67, 75, 79] provide better performance than PFSS. For file sharing systems in [71, 74], TTPs update the group key. Since they have powerful hardware configurations than client computers, we assume that TTPs are more scalable for using large groups.

As a consequence, PFSS together with KAP-PBC, provide better security properties than using group managers or TTPs. Moreover, our approach provides scalable group key computation for groups around 100 participants.

4.6. Discussions

Dynamic group key agreement protocols are expected to be used in applications such as file sharing systems. However, there are numbers of problems on the use of existing dynamic group key agreement protocols in file sharing systems such as the lack of privacy, the violation of availability and the dependency for key escrow.

In this chapter, we have proposed a key agreement protocol with partial backward confidentiality, called KAP-PBC. The proposed protocol provides a new security property called the partial backward confidentiality to overcome the problems of using group key agreement protocols as a security mechanism in file sharing systems. Moreover, KAP-PBC provides basic security properties of group key agreement protocols. Furthermore, security analysis of KAP-PBC against impersonation, eavesdropping and replay attacks is given. In addition, we have proposed the performance analysis of KAP-PBC regarding the communications cost and the computational cost. Performance analysis of KAP-PBC also shows that dynamic group operations provide efficient performance regarding the normal execution of the protocol.

Finally, we have proposed the proof of concept case study called PFSS in order to show the applicability of KAP-PBC. We have simulated PFSS regarding KAP-PBC service and a group update service. Simulation results show that PFSS is applicable for groups up to reasonable number of participants such as a hundred participants. Moreover, we have compared the security of PFSS with the existing file sharing systems. According to the comparisons, PFSS is the only system that provides all of the important security properties that we have addressed in this paper. Furthermore, we have addressed the implementation issues of PFSS as a real-life application.

5. GKAP-MANET: GROUP KEY AGREEMENT PROTOCOL FOR MOBILE AD HOC NETWORKS

Mobile ad hoc networks (MANETs) have numerous application areas that span from file sharing applications to vehicle-to-vehicle communication networks. Since entities in MANETs are mobile, providing secure communications among participants are significant issue. To overcome this issue, dynamic group key agreement protocols are used since MANETs are infrastructure-less, decentralized and mobile networks.

MANETs are formed by a combination of clusters. Therefore, communications of participants in MANET are categorized as in-cluster and inter-cluster communications. The first one is the communication of participants that are the members of the same cluster. The second one is the communication of participants that are not the members of the same cluster. In order to organize secure communication for such cluster-based network, most of the existing secure communication protocols use two levels security approach [10, 21, 22]. In the two-level security approach, different group key agreement protocols are used for in-cluster communications and inter-cluster communications. Cluster heads become the responsible node for decrypting/encrypting the incoming/outgoing messages for inter-cluster communications and in-cluster communications. However, using such approach may affect the security and energy consumption of cluster head. In terms security, availability of cluster members can be violated, because, cluster heads are single-point-of-failure for two-level security approach. Since, cluster heads perform more operations than other participants in their cluster, their batteries deplete faster. Dynamic group key agreement protocols can provide a better solution to overcome violation of availability and faster battery deplation problems. Since one group key is used, cluster heads do not need to perform more operations while organizing inter-cluster communications.

On the other hand, a secure cluster head selection is another significant issue since existing protocols do not provide such selection in a secure manner. The general approach for cluster head selection is that each participant publicly announces the number of connections

with other participants. Then, one with the maximum number of connection is selected as the cluster head. However, a malicious participant can claim that it has the highest number of connections to be the cluster head. Then, this malicious participant controls all the inter-cluster communication. A solution for this problem is to announce the list of the connected participants with the number of connections in a secure manner, which is one of our main motivations in this study.

In this chapter, we propose a group key agreement protocol that is adaptable for cluster-based communications in MANETs. Our contributions in this study are listed below:

(i) We propose a secure and efficient Group Key Agreement Protocol for MANETs, called GKAP-MANET, by improving the protocol in [17]. One of these improvement is removing the security vulnerabilities given in [18]. Also, the merge operation is added to the protocol to increase the adaptability for MANETs.

(ii) GKAP-MANET contains a new secure cluster head selection mechanism to overcome the malicious attempts of participants to be used for the computation of a new group key.

(iii) Better performance is provided in terms of reducing the communications and computation cost of group key computation during the execution of the protocol.

(iv) We present a set of simulations for an example GKAP-MANET application scenario.

The rest of the chapter is organized as follows. Definitions are given in the next section. In Section 5.2, we overview the important properties of group key agreement protocols regarding MANETs. In Section 5.3, we introduce the GKAP-MANET. Performance analysis is given in Section 5.4. GKAP-MANET application scenario and simulations are given in Section 5.5. Security of GKAP-MANET is analyzed in Section 5.6. Finally, we give discussion on the chapter in Section 5.7.

5.1. Comparison of Group Key Agreement Protocols on MANETs

In this section, we give the comparison of previously proposed group key agreement protocols in MANETs and the GKAP-MANET as shown in Table 5.1. Criteria used for comparing protocol properties are listed as follows:

(i) Authentication (Auth): In order to satisfy authentication property, a group key agreement protocol has to provide mechanism for participants to confirm the identity of any participants.

(ii) Security against Passive Attacks (SPA): Security of the protocol against eavesdropping of communications among participants during the computation of group keys.

(iii) Fault-Tolerance: This property is used for detecting the attempts of malicious participants for disrupting the computation of group key [6].

(iv) Forward Secrecy: If a group key agreement protocol provides forward secrecy, then there exist a protection against the compromise of produced group keys, such as the compromise of participant's long term key [15, 17].

(v) Dynamic Group Capability (DGC): Whether or not the group key agreement protocol provides auxiliary operations for handling participant joins and leaves.

(vi) Forward Confidentiality: Subsequent group keys cannot be obtained by participants who left the conference session [16].

(vii) Backward Confidentiality: Former group keys cannot be obtained by participants who joined to the conference session [16].

(viii) Secure Cluster Head Selection (SCHS): In MANETs, communications among participants is realized by in-cluster and inter-cluster communications

Table 5.1. Comparison for Security and Performance Properties of Group Key Agreement Protocols in MANETs

Protocols	**Auth.**	**SPA**	**Fault Tolerance**	**Forward Secrecy**	**DGC**	**Forward Confidentiality**	**Backward Confidentiality**	**SCHS**
CST-based Protocol [22]	✓	✗	✗	✗	✗	**NA**	**NA**	✗
DLST-based Protocol [22]	✓	✗	✗	✗	✗	**NA**	**NA**	✗
AHP-1 in [123]	✓	✓	✓	✓	✓	✓	✓	✗
AHP-2 in [123]	✓	✓	✓	✓	✓	✓	✓	✗
Protocol in [110]	✓	✓	✓	✓	✓	✓	✓	✗
Protocol in [124]	✓	✓	✓	✓	✗	**NA**	**NA**	✗
$A^2 - MAKE$ in [125]	✓	✓	✓	✓	✓	✓	✓	✗
Protocol in [126]	✓	✓	✓	✗	✓	**NA**	**NA**	✗
Protocol in [64]	✓	✓	✓	✗	✗	**NA**	**NA**	**NA**
AFTD in [127]	✓	✓	✓	✓	✓	✓	✓	**NA**
AUTH-CRTDH in [128]	✓	✓	✓	✓	✓	✓	✓	✗
AGDH in [129]	✓	✓	✓	✓	✓	✓	✓	✗
Protocol in [130]	✓	✓	✓	✗	✗	**NA**	**NA**	**NA**
GKAP-MANET	✓	✓	✓	✓	✓	✓	✓	✓

As shown in Table 5.1, GKAP-MANET is the only protocol that satisfies all criteria mentioned above for group key agreement protocols in MANETs. Moreover, GKAP-MANET is the first protocol that provides SCHS property. On the other hand, SCHS property is not considered in protocols [64, 130] and AFTD since they provide cluster-free communications for the participants in the group.

Another important property for group key agreement protocols in MANETs is the dynamic group capability. When new participant joins the group or an existing group leaves the group, the group key has to be updated. With dynamic group capability, a protocol can update the group key by using auxiliary operations without executing the whole protocol again. As far as the mobility of entities in MANETs is concerned, dynamic group capability becomes one of the most important properties of group key agreement protocols. Protocols in [22, 64, 124, 130] do not provide any auxiliary operations for dynamic groups. Since forward confidentiality and backward confidentiality properties are used only for protocols with dynamic group capabilities, they become Not Applicable (NA) for protocols [22, 64, 124, 130].

On the other hand, almost all of the protocols satisfy basic security properties of group key agreement protocols such as authentication, SPA, fault-tolerance and forward secrecy. However, CST-based protocol and DLST-based protocol in [22] do not provide SPA, fault tolerance and forward secrecy properties since the major concern in this study is to efficiently compute group key on centralized and distributed MANETs by using spanning tree. Since protocols presented in [22, 64, 126, 130] do not satisfy forward secrecy property, compromise of long-term key of any participant may endanger the security of previous and subsequent group keys.

In addition to the protocols given in Table 5.1, group key management approaches in [62, 63] are also important studies for key management in hierarchical MANETs. However, in both studies, proposed schemes concentrate on distributing keys by using a centralized authority instead of agreeing on a key by involving all of the participants. Therefore, protocols proposed in [62, 63] are out of scope for comparing with protocols listed in Table 5.1. Moreover, energy efficiency is another challenging issue in MANETs [131–134].

However, energy efficiency is not our primary concern and we analyze protocols regarding communications and computation costs.

5.2. General Definitions of Group Key Agreement Protocols for MANETs

In this section, we give the specific definitions for group key agreement protocols that operate on MANETs.

Definition 5.1. *Adjacent Participants. Let U_i and U_j be the two different participants in $\mathcal{U}$. If dist(U_i, U_j) $= 1$, these participants are called adjacent participants.*

Definition 5.2. *Neighborhood. For each participant $U_i \in \mathcal{U}$, the neighborhood of U_i is denoted as $N_{U_i} = \{U_1, U_2, \ldots, U_m\}$, where dist($U_i, U_j$) $= 1$, $\forall U_j \in N_{U_i}$, $N_{U_i} \subseteq \mathcal{U}$ and m is the number of participants in the neighborhood.*

Definition 5.3. *Cluster Head. Let U_i be a participant in $\mathcal{U}$ and N_{U_i} be its neighborhood. If U_i is the participant that has the maximum number of adjacent participants in its neighborhood, then this participant is called as cluster head.*

Definition 5.4. *Cluster. Each cluster head U_i together with its neighborhood N_{U_i} form a cluster.*

Definition 5.5. *Non-clustered Participants. Each participant $U_i \in \mathcal{U}$ that have distance dist(U_i, U_j) ≥ 2 for any cluster head participant $U_j \in \mathcal{U}$ is called non-clustered participant. This participants do not involve into the cluster-key computation. After the execution of merging clusters operation, they involve into the group key computation by using the join non-clustered participant operation. Details are given in the next section.*

5.3. A Group Key Agreement Protocol for MANETs

In this section, we introduce a secure and efficient group key agreement protocol for MANETs. The GKAP-MANET is based on the key agreement protocol of Tseng in [17]. The protocol provides the most important group key agreement properties such as authentication, fault-tolerance and forward secrecy by improving the non-authenticated protocol in [7]. Furthermore, detailed security analysis of the protocol against impersonation and passive attacks was given in [17]. However, in [18], an attack was proposed to show that Tseng's protocol does not provide backward confidentiality and forward confidentiality properties by using the vulnerabilities for joining and leaving operations. Therefore, we have modified the joining and leaving operations by adding the key renewal operation to provide backward and forward confidentiality properties. In addition, a merge operation is used for computing group key in order to combine in-cluster and inter-cluster communications.

5.3.1. GKAP-MANET Algorithm

Let $\mathcal{U} = \{U_1, U_2, \ldots, U_n\}$ be the set of participants. The activity diagram of GKAP-MANET in UML notation is shown in Figure 5.1. Functions of GKAP-MANET algorithm are given in the following definitions.

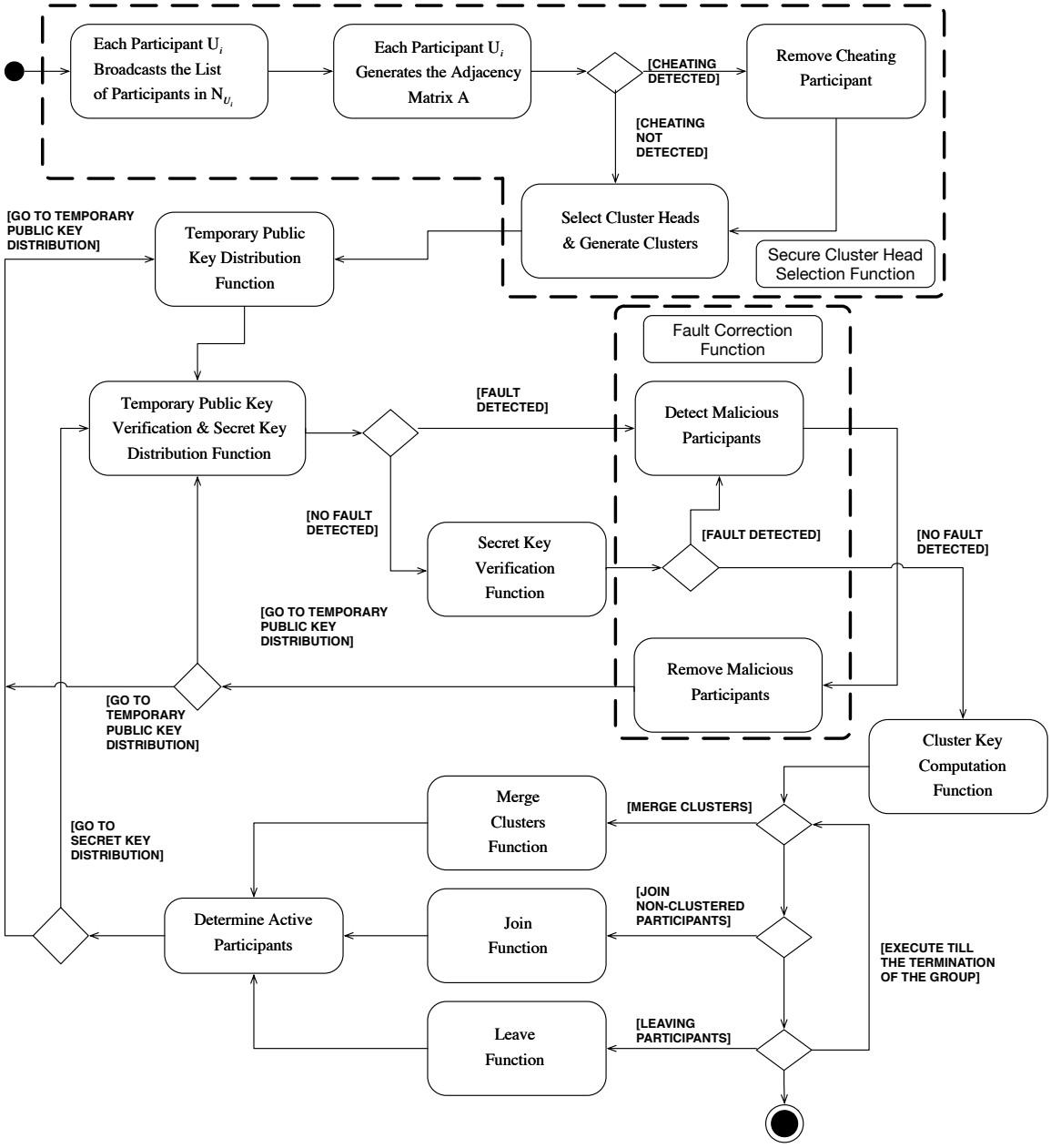

Figure 5.1. The Activity Diagram of GKAP-MANET in UML Notation

Definition 5.6. *Secure Cluster Head Selection Function. Each participant $U_i \in \mathcal{U}$ follows the steps in Figure 5.2.*

1 Each participant $U_i \in \mathcal{U}$ randomly selects $t_i \in Z_q^*$ and broadcasts the list of adjacent participants in its neighborhood as follows:

$$\gamma_i = g^{t_i} mod p,\ L_i = N_{U_i},\ \theta_i = t_i^{-1}(H(L_i, M) - \gamma_i x_i) mod q$$

2 Each participant $U_i \in \mathcal{U}$ verifies the incoming broadcast messages of each participant $U_j \in \mathcal{U}$ by using the equation $g^{H(\gamma_j, M)} = y_j^{\gamma_j} \gamma_j^{\theta_j}$, where $U_i \neq U_j$. Then, each U_i generates its own adjacency matrix $\mathcal{A}_{adj}^i$ by verifying incoming broadcast messages.

3 After the verification, if no cheating participant is detected, the participant with the maximum adjacent node in its neighbourhood is selected as the *cluster head*. In case of equality, the participant with the minimum ID is selected as the cluster head. For instance, let U_p, U_r and U_s with the order $p < r < s$ are the participants with the maximum adjacent node in their neighborhood. Then, U_p is selected as the cluster head.

4 According to the mismatch entries in Adjacency matrix $\mathcal{A}_{adj}^i$ of each participant U_i, cheating participants are discarded and each participants executes the function from Step 1.

5 After the cluster heads are selected, the participant set is divided into clusters according to these participants. Each cluster head U_i and their neighborhood N_{U_i} form a cluster such as $C_l = U_i \cup N_{U_i}$, where $l \in (1, 2, \ldots, s)$.

Figure 5.2. Secure Cluster Head Selection Function

Definition 5.7. *Temporary Public Key Distribution Function. After the cluster heads are selected, Each participant U_i generates and distributes temporary public key as shown in Figure 5.3.*

1 Each participant U_j randomly selects two short-term secret keys $k_j, v_j \in Z_q^*$.

2 Then, U_j computes the following parameters and broadcast them to the participant in its cluster:

$$\omega_j = g^{k_j} mod p,\ A_j = g^{v_j} mod p,\ B_j = v_j^{-1}(H(\omega_j, M) - A_j x_j) mod q$$

Figure 5.3. Temporary Public Key Distribution Function

Definition 5.8. *Temporary Public Key Verification and Secret Distribution Function. Each participant U_i verifies the temporary public keys of each participant in the same cluster. Then, generates and distributes secret key as given in Figure 5.4.*

1 After the temporary public keys are distributed, each participant $U_j \in C_l$, where C_l denotes cluster l for some positive integer l, verifies the broadcast messages for each participant $U_k \in C_l$, $(\omega_k, A_k, B_k, T_k)$ for $(1 \leqslant k \leqslant n, j \neq k)$ by using the equations $2 \leqslant \omega_k \leqslant p$, $g^{H(\omega_k, M)} = (y_k^{A_k} A_k^{B_k}) mod p$ and $\omega_k^q mod p \equiv 1$.

2 If all of the equations hold for any participant U_k, then U_j set verification matrix value as $V_{j,k} =$ "*success*".

3 Otherwise, U_i set verification matrix value as $V_{j,k} =$ "$failure$"

4 Each participant U_i executes Fault Correction Function.

5 U_j randomly selects $r_j \in Z_q^*$ for generating the secret key z_j and signature parameters. Then U_j broadcasts the following parameters to the other participants in its cluster:

$$\alpha_j = g_j^r mod p,\ \beta_j = (\omega_{j+1}/\omega_{j-1})^{r_j} mod p,\ \delta_j = r_j + H(z_j, \alpha_j, \beta_j, M) k_j mod q$$

Figure 5.4. Temporary Public Key Verification and Secret Distribution Function

Definition 5.9. *Secret Key Verification Function. Each participant U_i verifies the secret keys of broadcasted by other participant in the same cluster as shown in Figure 5.5.*

1 After the broadcast messages at 3 are exchanged by participant in the cluster, each U_j verifies the broadcast messages for each participant $U_k \in C_l$, $(z_k, \alpha_k, \beta_k, \delta_k)$ for $U_k \in$ by using the following equations:

$$g^{\delta_k} = \alpha_k \omega_k^{H(z_k,\alpha_k,\beta_k,\delta_k)} mod p,\ (\omega_{k+1}/\omega_{k-1})^{\delta_k} = \beta_k z_k^{H(z_k,\alpha_k,\beta_k,\delta_k)}$$

2 If all of the equations hold for any participant U_k, then U_j sets verification matrix value as $V_{j,k} =$ "*success*".

3 Otherwise, U_i set verification matrix value as $V_{j,k} =$ "$failure$"

4 Each participant U_i executes Fault Correction Function.

Figure 5.5. Secret Key Verification Function

Definition 5.10. *Fault Correction Function. If any malicious attempt is detected, then the fault correction function is executed as shown in Figure 5.6.*

1 Each participant U_j checks for the message $V_{i,m} =$ "$failure$" for U_m, where $m \neq i \neq j$.

2 According to the Function that fault was detected, U_j re-verifies the broadcast message of U_m. If fault is detected again, U_m is discarded from the cluster-key communication and marked as "malicious participant" and Leave Function is executed for U_m.

3 Otherwise, U_m is the *honest participant* and U_i tries to disrupt the true computation of the cluster-key. Therefore, U_i is discarded from the cluster-key and Leave Function is executed for U_i.

Figure 5.6. Fault Correction Function

Definition 5.11. *Cluster Key Computation Function. After the temporary public keys and secret keys are detected, if no malicious attempt is detected, then the cluster key computation function is executed as shown in Figure 5.7.*

1 Each $U_i \in C_k$ calculates the key for each $C_i = \{U_1, U_2, \ldots, U_m\}$ is as follows:

$$K = \omega_{i+1}^{mk_i} \cdot z_i^{m-1} \cdot z_{i+1}^{n-2} \cdots z_{i-2} mod p \quad (5.1)$$

$$= g^{k_1k_2+k_2k_3+k_3k_4+\ldots+k_mk_1} mod p \quad (5.2)$$

Figure 5.7. Cluster Key Computation Function

Definition 5.12. *Merging Clusters Function. After keys of each clusters are computed, for clusters $C_1, C_2, \ldots, C_s$ the merging clusters function operates as shown in Figure 5.8.*

1 If participant $U_i \in C_l$, where $l \in (1, 2, \ldots, s)$, where s is the number of clusters in MANET, is the last participant of any cluster, then U_i executes GKAP-MANET from Temporary Public Key Distribution Function to Secret Key Verification Function according to U_j, where $U_j \notin C_l$ and U_j is the first participant in C_{l+1}.

2 If participant $U_i \in C_l$, where $l \in (1, 2, \ldots, s)$, is the participant before the last participant in any cluster, then U_i executes GKAP-MANET from Temporary Public Key Distribution Function to Secret Key Verification Function.

3 Otherwise, U_i broadcast the previously calculated parameters and verifies the incoming messages.

4 If no fault is detected, new cluster is generated by merging. Then, all participants in the group execute Cluster Key Computation for the new cluster.

Figure 5.8. Merge Clusters Function

Definition 5.13. *Join Function. After the clusters have been merged, the non-clustered participants join the group. Moreover, this function is also used for adding new participants after the group key is computed. Let $\mathcal{U}' = \{U_1, U_2, \ldots, U_m\}$ be the participant set after the execution of Merging Clusters Function and let $U_{m+1}, U_{m+2}, U_{m+3}, \ldots,$ U_{m+k} be the non-clustered participants, Details of the join function are given in Figure 5.9.*

1 The last participant U_m and the new participants $U_{m+1}, U_{m+2}, U_{m+3}, \ldots, U_{m+k}$ executes GKAP-MANET from Temporary Public Key Computation Function to Secret Key Verification Function.

2 U_1 and U_{m-1} execute GKAP-MANET from the Temporary Public Key Verification & Secret Key Distribution Function.

3 The rest of the participants broadcast previously generated temporary keys and secret keys.

4 If no fault is detected, all participants in the group execute Cluster Key Computation.

5 If cluster formations is updated and cluster head selections has to be repeated.

Figure 5.9. Join Function.

Definition 5.14. Leave Function. *If a participant leaves the group communication, this function is executed. Let U_i be the leaving participant then the leave function is given in Figure 5.10.*

1 If $U_i \in C_l$ is the cluster head, each participant in C_l execute Secure Cluster Head Selection Function for selecting the new cluster head.

2 U_{i-1} executes GKAP-MANET from Temporary Public Key Distribution Function to Secret Key Verification Function.

3 U_{i-2} and U_{i+1} execute GKAP-MANET from Temporary Public Key Verification & Secret Key Distribution Function to Secret Key Verification Function.

4 If no fault is detected, then all of the participants execute Cluster Key Computation Function.

Figure 5.10. Leaving Participants Function.

5.3.2. Effects of Mobility in GKAP-MANET

In MANETs, cluster formation changes when the participants move. There are three possible cases for the mobility of participants in GKAP-MANET:

(i) Participant Joins Group: After the group key is computed by all of the participants, new participant(s) joins the group due to their mobility. Then, they are treated as non-clustered participants and Join Function is executed in order to update the group key.

(ii) Participant Leaves Group: When participants leave due to their mobility, then Leave Function is executed.

(iii) Participant Moves within Group: Since participants in MANETs are mobile, they can move from one cluster to another while moving within a group and the group topology changes. Although there is no need to re-compute the group key, cluster head selection need to be repeated. Therefore, each participant U_i periodically checks for the participants in its neighbourhood. If a new participant is detected or an existing participant cannot be reached, the Cluster Head Selection is repeated for selecting the new cluster head.

5.3.3. Boundary Conditions of GKAP-MANET

Let $\mathcal{U}$ be the set of participants and $n = |\mathcal{U}|$ be the number of participants in MANET. Then, boundary condition of GKAP-MANET is as follows:

(i) Case $n = 1$, Participant waits for other participants to start multi-party communication.

(ii) Case $n = 2$, In order to execute Cluster Key Computation of GKAP-MANET, the number of participant in $\mathcal{U}$ must be greater than or equal to three. Therefore, we define a variant of Diffie Hellman protocol [2] as GKAP-MANET-Alternate for $n = 2$ as shown in Figure 5.11:

(iii) Case $n \geq 3$, GKAP-MANET operates as defined in Section 5.3.1.

1 *Cluster Head Selection:* For $n = 2$, cluster head is selected according to the ID of a participant in the group. The one with the lowest ID is selected as the cluster head. Let U_i and U_j be the participants in $\mathcal{U}$, where $i \neq j$. Then, if $i < j$, U_i is selected as cluster head. Otherwise, U_j is selected as cluster head.

2 *Temporary Public Key Distribution:* Each participant U_j randomly selects two short-term secret keys $k_j, v_j \in Z_q^*$. Then, U_j computes the following parameters and broadcast them to the participant in its cluster:

$$\omega_j = g^{k_j} mod p,\ A_j = g^{v_j} mod p,\ B_j = v_j^{-1}(H(\omega_j, M) - A_j x_j) mod q$$

3 *Temporary Public Key Verification and Cluster Key Computation:* After the temporary public keys are distributed, each participant $U_j \in C_l$ verifies the broadcast messages for each participant $U_k \in C_l$, $(\omega_k, A_k, B_k, T_k)$ for $(1 \leqslant k \leqslant n, j \neq k)$ by using the equations $2 \leqslant \omega_k \leqslant p$, $g^{H(\omega_k, M)} = y_k^{A_k} A_k^{B_k} mod p$, and $\omega_k^q mod p \equiv 1$. If all of the equations hold for any participant U_k, then U_j set verification matrix value as $V_{j,k} =$ "*success*". Otherwise, U_i set verification matrix value as $V_{j,k} =$ "$failure$" and cancel the cluster key computation.

Figure 5.11. GKAP-MANET-Alternate.

5.4. Performance Analysis of GKAP-MANET

In this section, we analyze the performance of the GKAP-MANET regarding communications and computation costs. Moreover, we compare GKAP-MANET performance with existing group key agreement protocols in literature.

5.4.1. Communications Cost Analysis

In this section, we analyze the communications cost of GKAP-MANET with respect to the message length and the total number of communication rounds. Let $U_i \in \mathcal{U}$ be a participant in the execution of GKAP-MANET. Then, the message length and the total number

of communication rounds are denoted as $\mathcal{C}_{length}(U_i)$ and $\mathcal{C}_{round}(U_i)$, respectively. Moreover, the overall communication cost is denoted as $\mathcal{C}_{comm}(U_i) = \mathcal{C}_{length}(U_i) + \mathcal{C}_{round}(U_i)$.

In GKAP-MANET, both the message length and the number of communication rounds are fixed. During the computation of the cluster keys, each participant communicates three times with the other participants in the cluster for the cluster head selection, distributing the temporary keys and distributing the secret keys. On the other hand, the GKAP-MANET has to execute merging and joining operations to compute the group key for all participants in MANET and leaving operation to update the key if any participant leaves the MANET. The total number of communication rounds for merging, joining and leaving operations are as follows:

- For the cluster head selection, each participant U_i broadcasts L_i, γ_i, θ_i. Therefore,

$$\mathcal{C}_{round}(U_i) = 1$$

- For the cluster key computation, each participant U_i executes from Temporary Public Key Computation Function to Cluster Key Computation Function. If no fault is detected during the computation of cluster key, the total number of communication rounds per participant is

$$\mathcal{C}_{round}(U_i) = 2$$

- For the merging clusters operation, assume that $C_1 = \langle U_{1,1}, U_{1,2}, \ldots, U_{1,m} \rangle$ and $C_2 = \langle U_{2,1}, U_{2,2}, \ldots, U_{2,m} \rangle$ are list of participants in clusters C_1 and C_2. C_1 and C_2 are concatenated as $C_1 || C_2 = \langle U_{1,1}, U_{1,2}, \ldots, U_{1,n}, U_{2,1}, U_{2,2}, \ldots, U_{2,n} \rangle$. Then, participants $U_{1,n}$ and $U_{2,n}$ execute from Temporary Public Key Computation Function to Cluster Key Computation Function. Participants $U_{1,n-1}$ and $U_{2,n-1}$ execute from Temporary Public Key Verification & Secret Key Distribution Function to Cluster Key Computation

Function. The total number of communication rounds per participant is:

$$\mathcal{C}_{round}(U_{j,i}) = 2$$

where j is the index of clusters C_1, C_2 and i is the id of a participant in cluster C_j. Participants that are not in $U_{1,n}, U_{2,n}, U_{1,n-1}$ and $U_{2,n-1}$ has to broadcast existing parameter values.

- For the joining operation, assume that $\mathcal{C} = \langle U_1, U_2, \ldots, U_m \rangle$ is the MANET and U_{m+1} is the new participant that joining into the MANET. Then, U_m and U_{m+1} execute Temporary Public Key Computation Function to Cluster Key Computation Function and U_{m-1} execute Temporary Public Key Verification & Secret Key Distribution Function to Cluster Key Computation Function. The total communication rounds per participant is

$$\mathcal{C}_{round}(U_i) = 2$$

  participants that are not in U_{m-1}, U_m and U_{m+1} have to broadcast existing temporary public key.
- For the leaving operation, assume that $\mathcal{C} = \langle U_1, U_2, \ldots, U_m \rangle$ is the MANET and U_i is the leaving participant. Then, U_{i-1} executes from Temporary Public Key Computation Function to Cluster Key Computation Function and U_{i-2} executes from Temporary Public Key Verification & Secret Key Distribution Function to Cluster Key Computation Function. The total broadcasts for each leaving operation is 5.

$$\mathcal{C}_{round}(U_{i-1}) = 2$$

$$\mathcal{C}_{round}(U_{i-2}) = 1$$

Almost all of the equations in the broadcast messages are modular operations. Therefore, message length of a parameter is calculated as the cardinality of the bit representation

of the modular base. For instance, the length of the temporary public-key $\omega_i = g^{k_i} \ mod \ p$ is $|p|$ bits. In GKAP-MANET, the total length of broadcast messages for each participant in Temporary Public Key distribution and Secret Key distribution is

$$C_{length}(U_i) = 5|p| + 2|q| \ bits$$

Let m be the total number of participants in the group. The communication cost comparison in terms of message length for the cluster-key computation of GKAP-MANET and group key computation of protocols in [9], [16] and [14] is given in Table 5.2. Since there is no cluster head selection method in the existing protocols, comparisons only consist of the communications cost for Temporary Public Key distribution and Secret Key distribution of GKAP-MANET. According to the comparisons in Table 5.2, the message lengths of the existing protocols is dependent on the number of participants, m, in the group. In the cluster-key computation of GKAP-MANET, message length is independent from the number of participants in the group, which is $5|p| + 2|q|$.

Table 5.2. Comparison for the communications costs during the cluster-key computation

Protocol	$C_{length}(U_i)$
Protocol in [14]	$(m+2)\|q\| + 4\|p\|$
Protocol in [9]	$(m+1)\|q\| + 2\|p\|$
Protocol in [16]	$(m+2)\|q\| + 4\|p\|$
GKAP-MANET	$5\|p\| + 2\|q\|$

5.4.2. Computational Complexity Cost Analysis

In this section, we analyze the computational complexity cost of GKAP-MANET. As defined in Section 2.2.2., we only concentrate on the computational complexity cost of modular exponentiation operations, denoted as T_{exp}, since they are assumed to be the most time consuming operations throughout the protocol.

The total modular exponentiation for each participant U_i of the cluster-key computation is as follows:

$$\mathcal{C}_{comp} = O(1) * T_{exp} = O(1)T_{exp} \tag{5.3}$$

In terms of merging operation, the computational complexity cost is

$$\mathcal{C}_{comp} = O(k) * T_{exp} = O(k)T_{exp}, \tag{5.4}$$

where k is the number of clusters. Computational complexity cost of adding a single participant into a group is

$$\mathcal{C}_{comp} = O(1) * T_{exp} = O(1)T_{exp} \tag{5.5}$$

Let m be the total number of participants in the group, the comparison of the computational complexity cost for the cluster key computations of GKAP-MANET and group key computation of protocols [16, 114, 135, 136] are given in Table 5.3.

Table 5.3. Comparison for the computational complexity costs during the cluster-key computation

Protocol	$\mathcal{C}_{comp}(U_i)$
Protocol in [16]	$O(m) \cdot T_{exp}$
Protocol in [135]	$\leqslant O(\log_3 m) \cdot T_{exp}$
Protocol in [114]	$\leqslant O(\log_2 m) \cdot T_{exp}$
Protocol in [136]	$O(\log_2 m) \cdot T_{exp}$
GKAP-MANET	$O(1) \cdot T_{exp}$

5.4.3. Overall Comments on Performance Analysis

Table 5.2 and Table 5.3 show that GKAP-MANET has better performance than other protocols regarding communication and computational complexity costs. Computational

complexity cost per participant increases logarithmically according to total number of participants, m, in existing protocols. On the other hand, the computational complexity cost per participant in GKAP-MANET is independent from the total number of participants and it depends only to modular exponentiations, $O(1)T_{exp}$. Thus, the GKAP-MANET has better performance in terms of computational complexity cost for MANETs. Moreover, the communications cost has also better performance than the existing protocols. According to the comparisons of communications costs in Table 3 the length of the messages exchanged during the key computation is linearly dependent on the number of participants, m, in the group. For GKAP-MANET, the message length is also independent from the number of participants. Since GKAP-MANET provides better performance in terms of communications and computational complexity costs, we expect to see more efficient energy consumption in a real world application.

5.5. Security Analysis of GKAP-MANET

In this section, we analyze the security of GKAP-MANET, which is an improved version of Tseng's group key agreement protocol [17]. Therefore, we first show that GKAP-MANET provides a better security level than Tseng's protocol does. Additionally, GKAP-MANET is secure against possible attacks on Tseng's algorithm [18]. Moreover, we show that the GKAP-MANET provides a secure cluster head selection mechanism. GKAP-MANET differs from Tseng's protocol in the following additional operations:

(i) Secure cluster head selection,
(ii) Merge operation,
(iii) Secure join and leave operations against attacks in [18].

On the other hand, GKAP-MANET follows exactly the same steps in [17] for cluster key computation. Therefore, the cluster key computation of GKAP-MANET provides security against impersonation, eavesdropping and replay attacks.

5.5.1. Authentication, Fault-Tolerance and Forward Secrecy

It is shown that the protocol in [17] provides authentication, fault-tolerance and forward secrecy properties. In this section, we show that GKAP-MANET protocol also provides authentication, fault-tolerance and additionally forward secrecy properties.

The authentication property in group key agreement protocols is used to detect whether the participant is a member of the group or not. This detection process is generally realized during the verification of messages broadcasted by participants in the group (or cluster in our case). In GKAP-MANET, participants communicate with each other while executing the following functions:

- Secure Cluster Head Selection Function: Each participant $U_i \in \mathcal{U}$ broadcasts the list of participants in its neighborhood. Signature paramaters for this step are γ_i and θ_i.
- Temporary Public Key Distribution Function: Each participant $U_i \in \mathcal{U}$. Each U_i broadcasts signatures A_i and B_j for temporarily public key ω_i.
- Temporary Public Key Verification and Secret Key Distribution Function: Each participant $U_i \in \mathcal{U}$ broadcasts it's secret key with signature parameters α_i, β_i and γ_i.

When a broadcast message is sent by participant U_i in any of the functions above, the transmitted parameters are verified by other participants in the group (or cluster). Since each of the signature parameters are signed by the long-term public keys of the participants and each participant has information about public keys of any other participant in the group, our protocol satisfies the authentication property. Moreover, for the dynamic group operations, merging clusters, join and leave, properties are also satisfied by GKAP-MANET because the same broadcast steps expressed above are used.

Furhermore, fault-tolerance property is used for eliminating malicious participants from the cluster key or group key computation. Fault-tolerance property consists of two actions:

- Fault detection: In GKAP-MANET, fault detection is realized in Step 2 of Secure Cluster Head Selection Function, in Step 1 of Temporary Public Key Verification & Secret Key Distribution Function and in Step 1 of Secret Key Verification Function. When a malicious participant tries to disrupt the execution of GKAP-MANET, it is marked as possible malicious participants by using the verification matrix, V.
- Fault correction: In GKAP-MANET, fault correction is realized in Fault Correction Function. Each participant U_i verifies the broadcast messages of participants where $V_{j,k} =$ "$failure$" and $i \neq j \neq k$ for any possible malicious participant U_k. According to the verification result, if U_k is malicious participant then it is removed from the group/cluster. Otherwise, U_j is removed from the group/cluster.

Thus, malicious attempts of participant can easily be detected and owner of malicious attempts are removed from the communication. Thus, GKAP-MANET provides fault-tolerance property.

The last basic security property of group key agreement protocols is the forward secrecy. The forward secrecy property is used for providing a mechanism to prevent against compromise of group keys. Since the temporary keys are used in the GKAP-MANET, the forward secrecy is also satisfied as in [17].

5.5.2. Security of Dynamic Group Operations

It is shown that the Tseng's protocol does not provide backward and forward confidentiality properties against the attack [18]. In this section, we show that GKAP-MANET achieve backward confidentiality in merge and join operations. Moreover, we show that forward confidentiality is also satisfied for leave operation. In addition, we define key freshness of GKAP-MANET based on the backward confidentiality and forward confidentiality properties. Following lemmas and corresponding theorem show that GKAP-MANET is secure against the attack proposed in [18]. While proving the lemmas and theorem, we benefit from the difficulty of discrete logarithm problem. Since p is a large prime number, q is the prime factor of $p-1$ and g is a generator for subgroup G_q, by the definition of discrete logarithm problem, it is infeasible to obtain a from $b = g^a mod p$, where a and b any random positive

integers in Z_q.

Lemma 5.1. *Under the difficulty of discrete logarithm problem, the merging clusters provides backward confidentiality.*

Proof. Assume that $C_1 = \{U_1, U_2, \ldots, U_m\}$ and $C_2 = \{U_{m+1}, U_{m+2}, \ldots, U_{2m}\}$ are two clusters to be merged. Let

$$K_1 = g^{k_1k_2+k_2k_3+\ldots+k_mk_1} mod\, p$$

and

$$K_2 = g^{k_{m+1}k_{m+2}+k_{m+2}k_{m+3}+\ldots+k_{2m}k_{m+1}}$$

be the cluster keys, respectively. Since the last participants re-execute the protocol from Temporary Key Distribution Function, the group key after merging operation will be as follows:

$$K' = g^{k_1k_2+k_2k_3+\ldots+k'_mk_{m+1}+k_{m+1}k_{m+2}+k_{m+2}k_{m+3}+\ldots+k'_{2m}k_1}$$

Therefore, it is infeasible to compute cluster keys by using the group key under the difficulty of discrete logarithm problem. Hence the merging operation satisfies the backward confidentiality. □

Lemma 5.2. *Under the difficulty of discrete logarithm problem, the joining non-clustered participants operation provides backward confidentiality.*

Proof. Let $C = \{U_1, U_2, \ldots, U_m\}$ be the merged clusters and $K = g^{k_1k_2+k_2k_3+\ldots\ +k_mk_1}\ mod\, p$ be the cluster key. Assume that U_{m+1} is the non-clustered participant to be added. After the joining operation, the group key is

$$K = g^{k_1k_2+k_2k_3+\ldots+k'_mk_{m+1}+k_{m+1}k_1} mod\, p$$

Since the last participant, U_m, re-execute the protocol from Temporary Key Distribution Function, the temporary key value for that participant will be changed. Therefore, it is infeasible for U_{m+1} to compute group key by using the new group key under the difficulty of discrete logarithm problem. Hence the joining non-clustered participants operation satisfies the backward confidentiality. □

Lemma 5.3. *Under the difficulty of discrete logarithm problem, the leave operation provides forward confidentiality.*

Proof. Assume that $C = \{U_1, U_2, \ldots, U_m\}$ is the merged clusters and the cluster key is

$$K = g^{k_1k_2+k_2k_3+\ldots+k_{i-1}k_i+k_ik_{i+1}+\cdots+k_mk_1} mod\, p$$

Assume that U_i is the leaving participant. After the leaving operation, the group key is as follows:

$$K = g^{k_1k_2+k_2k_3+\ldots+k_{i-2}k'_{i-1}+k'_{i-1}k_{i+1}+\ldots+k'_mk_{m+1}+k_{m+1}k_1} mod\, p$$

Since the participant U_{i-1} re-execute the protocol from Temporary Key Distribution Function, the temporary key value for that participant will be changed. Therefore, it is infeasible for U_i to compute the new group key by using the old group key under the difficulty of discrete logarithm problem. Hence the leaving participants operation satisfies the forward confidentiality. □

Theorem 5.1. *Under the difficulty of discrete logarithm problem, the key updated by merging, joining and leaving operations is always fresh.*

Proof. Following from lemmas 5.2 and 5.3 the backward confidentiality is satisfied for merging and joining operations under the difficulty of discrete logarithm problem. In addition, the forward confidentiality is satisfied for leaving operation under the difficulty of discrete logarithm problem. Hence, the key modified by these operations is always fresh. □

5.5.3. Analysis of Secure Cluster Head Selection

In this section, we show that GKAP-MANET provides secure cluster head selection. In previous studies, each participant in MANET publicly broadcasts number of adjacent nodes in its neighborhood. Then, the one with the maximum number of adjacent nodes is selected as cluster head. However, previous studies do not provide secure cluster head selection. In GKAP-MANET, the following cases of secure cluster head selection exist:

(i.) A malicious participant, who does not have the maximum number of adjacent nodes in her neighborhood, can claim that she is the cluster head.

(ii.) A malicious participant can impersonate as a participant, who has the maximum number of adjacent nodes in his neighborhood.

Table 5.4. Toy Example

Participant	L_i
U_1	U_2, U_3
U_2	U_1, U_3, U_{18}
U_3	U_1, U_2, U_4, U_5, U_6
U_4	U_3
U_5	U_3
U_6	U_3

For case (i): Assume that Table 5.4 show the list of adjacent participants for $C_1 = \{U_1, U_2, U_3, U_4, U_5, U_6\}$. Therefore, the Adjacency matrix $\mathcal{A}^i_{adj}$ generated by each participant

U_i in C_1 is as follows:

$$\mathcal{A}^i_{adj} = \begin{array}{c} \\ U_1 \\ U_2 \\ U_3 \\ U_4 \\ U_5 \\ U_6 \end{array} \begin{array}{c} \begin{array}{cccccccc} U_1 & U_2 & U_3 & U_4 & U_5 & U_6 & \dots & U_{18} \end{array} \\ \begin{pmatrix} 0 & 1 & 1 & 0 & 0 & 0 & \dots & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & \dots & 1 \\ 1 & 1 & 0 & 1 & 1 & 1 & \dots & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & \dots & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & \dots & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & \dots & 0 \end{pmatrix} \end{array}$$

Let U_2 be the malicious participant, in order to be selected as cluster head, U_2 has to have 6 connections. However, each participant broadcast adjacent participant as a list. Therefore, each participant can generate adjacency matrix and easily detects that U_2 is cheating. The following theorem and its proof show the formal version of the analysis above:

Theorem 5.2. *If all of the participants in $\mathcal{U} = \{U_1, U_2, U_3, \dots, U_n\}$ fully follows the Secure Cluster Head Selection Function, then cluster head is correctly selected by each participant U_i.*

Proof. Let U_i be the malicious participant. To be selected as the cluster head, U_i broadcasts L_i filled with not real adjacent nodes. Then, each $U_j \in \mathcal{U}$, where $U_j \neq U_i$ verifies the broadcast messages $(\gamma_k, L_k, \theta_k)$ of every other participant $U_k \in \mathcal{U}$ in the neigborhood. Since each γ_k, $H(L_k, M) \in Z_q$ are unique, the received list L_k of each participant must be unique for all participants. Therefore, each U_j can verify any connection between U_i and U_t, where $j \neq i \neq t$, by using L_i and L_t. Thus, if participants fully follow the Secure Cluster Head Selection Function, then the cluster head is correctly selected. Otherwise, malicious participants are detected by using the adjacency matrix of each participant and they are removed before the cluster key computation. □

For case (ii): Since L_i parameter is broadcasted without applying any encryption, a malicious participant can impersonate as U_3 and claim that she has the maximum number of adjacency nodes in her neighborhood. In order to prove security against impersonation at-

tack, we adopt the random oracle model assumption [99]. The assumption of random oracle is that the one-way hash functions are accepted as true random functions. In this analysis, we assume that the broadcast message parameters for cluster head selection, $(\gamma_i, L_i, \theta_i)$, are existentially un-forgeable. Under the random oracle model, $H(L_i, M)$ is an independent random variable from (L_i, M). Let malicious participant, U_2, interrupts broadcast message of U_3 and change the L_3 value with L_2. In order to sign L_2 with the long term private key of U_3, U_2 has to obtain x_3 from $\theta_3 = t_3^{-1}(H(L_3) - \gamma_3 x_3) mod q$, which is as hard as solving discrete logarithm problem. Therefore, cluster head selection of GKAP-MANET is secure against impersonation attack.

The final security issue regarding the secure cluster head selection is the violation of availability if cluster heads are single-point-of-failure in the group. For some reason, if one of the cluster heads leaves the communication, GKAP-MANET provides protection against violation of availability as follows:

- As given in the GKAP-MANET algorithm, when a cluster head leaves communication due to the mobility, the Secure Cluster Head Selection Function is triggered. Therefore, if a cluster head becomes out of service, a new cluster head is automatically selected.
- Since all of the participants in the group use the same group key for both in-cluster and inter-cluster connection, change of cluster head does not bring any extra computation and communications costs. All the participants in the group execute Leave Function of GKAP-MANET for updating the group key.

5.5.4. Overall Comments on Security Analysis

As shown in the security analysis, GKAP-MANET provides better security level than Tseng's protocol. In addition, the vulnerability of Tseng's protocol in [17] that results from the joining and leaving operations is resolved. Moreover, this section shows that all of the security properties are satisfied. Finally, it is shown that cluster head selection of GKAP-MANET is secure against cheating attempts and impersonation attacks of malicious participants. Furthermore, GKAP-MANET also provides protection against violation of availability.

5.6. Simulations of GKAP-MANET

In this section, we give the simulations of GKAP-MANET regarding computation and communications costs. First of all, we illustrate the execution of GKAP-MANET on an example network. Then, we simulate GKAP-MANET for the given network.

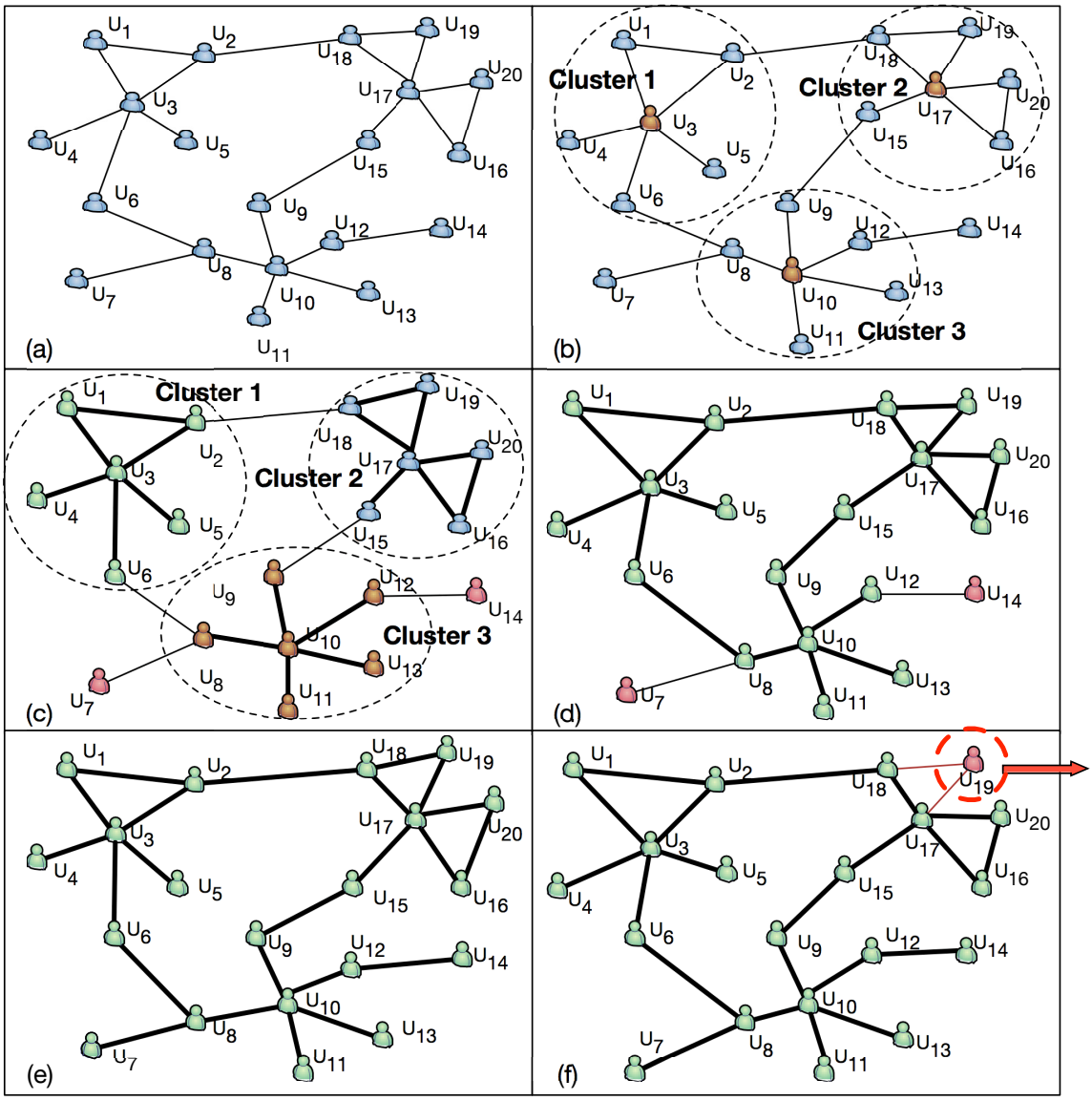

Figure 5.12. Execution of GKAP-MANET on an Example Application Scenario

5.6.1. GKAP-MANET Example Application Scenario

For the execution of GKAP-MANET, the example network is given in Figure 5.12 (a)-(f). Let $\mathcal{U} = \{U_1, U_2, U_3, \ldots, U_{20}\}$ be the set of participants. Let weak lines among participants show peer-to-peer connection and strong lines show secure communication among participants. The execution of GKAP-MANET is as follows:

- Figure 5.12(a) represents the initial state of the participants before the Secure Cluster Head Selection Function of the protocol. First, each participant in $U_i \in \mathcal{U}$ broadcasts the number of adjacent participants as shown in the example MANET.
- In Figure 5.12(b) shows the cluster head selection. Each participant U_i compare incoming messages from the neighborhood with its own message and form the adjacency matrix $\mathcal{A}^i_{adj}$. If U_i has the maximum number of adjacent nodes in the neighborhood, then U_i claim itself as the cluster head. As shown in Figure 2b, the participants U_3, U_{10} and U_{17} are the cluster heads, because, they have the maximum number of adjacent nodes in their neighborhood. Let $\mathcal{S} = \{U_3, U_{10}, U_{17}\}$ be the set of cluster heads, clusters are formed as follows. For each $U_i \notin \mathcal{S}$, if $dist(U_i, U_j) = 1$ for $U_j \in \mathcal{S}$, U_i is located in the cluster of U_j. If a node U_k is not in the neighborhood of any cluster head, then U_k waits till the execution of Join Function. Let C_1, C_2 and C_3 be the sets of participants for Cluster 1, Cluster 2 and Cluster 3, respectively. Then, according to the Figure 2b, clusters contains the following participants: $C_1 = \{U_1, U_2, U_3, U_4, U_5, U_6\}$, $C_2 = \{U_{15}, U_{16}, U_{17}, U_{18}, U_{19}, U_{20}\}$ and $C_3 = \{U_8, U_9, U_{10}, U_{11}, U_{12}, U_{13}\}$.
- Figure 5.12(c) shows the organization of the clusters after the execution of Cluster-Key Computation Function. Since the sets of participants for each cluster is determined, the cluster-key computation can be realized from Temporary Key Computation to Cluster Key Computation for the circular participant lists of clusters $\mathcal{C}_1 = \langle U_1, U_2, U_3, U_4, U_5, U_6 \rangle$, $\mathcal{C}_2 = \langle U_{15}, U_{16}, U_{17}, U_{18}, U_{19}, U_{20} \rangle$, and $\mathcal{C}_3 = \langle U_8, U_9, U_{10}, U_{11}, U_{12}, U_{13} \rangle$.
- Figure 5.12(d) represents the secure communication lines after the execution of Merging Clusters Function. The physical connections of clusters are as follows: for C_1 and C_2, U_2 and U_{18} are adjacent participants, for C_1 and C_3, U_6 and U_9 are adjacent participants and for C_2 and C_3, U_{15} and U_9 are the adjacent nodes. However, we are

not dealing with the physical communication lines while executing the protocol since MANET nodes have multihop capability. According to the Merging Clusters Function, lists are concatenated while merging. First, $\mathcal{C}_1$ and $\mathcal{C}_2$ are concatenated as $\mathcal{C}_1||\mathcal{C}_2$, after that the $\mathcal{C}_3$ is concatenated to $\mathcal{C}_1||\mathcal{C}_2$. The resulting list after the execution of Merging Clusters Function becomes $\mathcal{C}_1||\mathcal{C}_2||\mathcal{C}_3 = \langle U_1, U_2, U_3, U_4, U_5, U_6, U_{15}, U_{16}, U_{17}, U_{18}, U_8, U_9, U_{10}, U_{11}, U_{12}, U_{13} \rangle$.

While concatenating lists in Merging Clusters Function, for $\mathcal{C}_1||\mathcal{C}_2$, U_6 executes the protocol from Temporary Public Key Distribution Function to Cluster Key Computation Function and U_5 executes the protocol from Temporary Public Key Verification & Secret Key Distribution Function to Cluster Key Computation Function and for $\mathcal{C}_1||\mathcal{C}_2||\mathcal{C}_3$, U_{20} executes the protocol from Temporary Public Key Distribution Function to Cluster Key Computation Function and U_{19} executes the protocol from Temporary Public Key Verification & Secret Key Distribution Function to Cluster Key Computation Function. As shown in the example, each merging operation has a constant computation overhead.

- In Figure 5.12(e), non-clustered participants are added into the group communication of MANET. As we stated in the explanation of Figure 2d, non-clustered participants, U_7 and U_{14} are added at the end of the list, respectively. The resulting list of MANET becomes:

$$\mathcal{L} = \langle U_1, U_2, U_3, U_4, U_5, U_6, U_{15}, U_{16}, U_{17}, U_{18}, U_8, U_9, U_{10}, U_{11}, U_{12}, U_{13}, U_7, U_{14} \rangle$$

Details of the operation are given in the definition of Join Function. While adding U_7 into the list, U_{20} executes the protocol from Temporary Public Key Distribution Function to Cluster Key Computation Function and U_{19} executes the protocol from Temporary Public Key Verification & Secret Key Distribution Function to Cluster Key Computation Function and for U_{14}, U_7 executes the protocol from Temporary Public Key Distribution Function to Cluster Key Computation Function and U_{20} executes the protocol from Temporary Public Key Verification & Secret Key Distribution Function to Cluster Key Computation Function. As shown in the example each add operation has also constant computation overhead.

- In addition to Merge clusters and joining non-clustered participants operations, it is possible to handle leaving participants from MANET as well. As defined in Leave Function, any participant can leave the communication or any participant can try to cheat the other participants during the execution of group key. In Figure 5.12(f), possible leaving operation is shown. Like the other operations that we defined for Figure 2d and 2e, physical communication is not our primary concern during the execution of the protocol. In a formal representation, if U_{19} leaves the group communication, then U_{18} executes the protocol from Temporary Public Key Distribution Function to Cluster Key Computation Function and U_{17} executes the protocol from Temporary Public Key Verification & Secret Key Distribution Function to Cluster Key Computation Function. Thus, it is clear that the leaving participant step of the protocol also has the constant computation cost.

5.6.2. Simulations of GKAP-MANET on Example Scenario

In this section, we give the simulations of GKAP-MANET based on the example scenario on in Figure 5.13. Simulations were carried out by using Python 2.7. that run on a MacBook Air 2012 early release with 250 GB SSD disk, 1.8GHz Intel Core i5 processor and 4GB 1600 MHz DDR3 RAM.

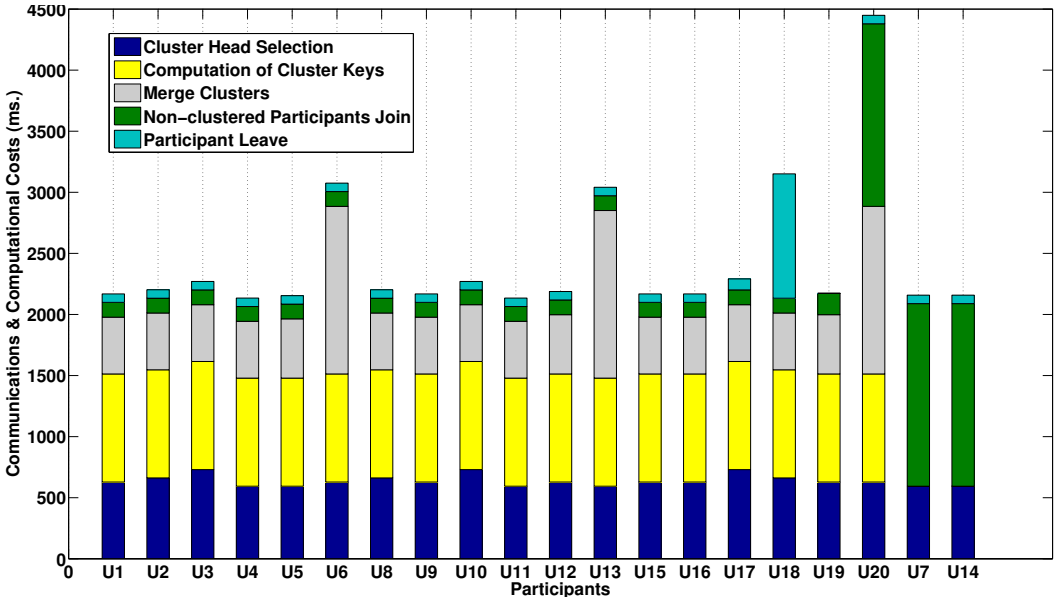

Figure 5.13. Simulation of GKAP-MANET on the example scenario in Figure 2. Participants U_7 and U_{14} are nonclustered participants.

As defined in the previous section, the set of participants in the MANET is $\mathcal{U} = \{U_1, U_2, U_3, \ldots, U_{20}\}$. In order to show the communications and computation costs of participants more clearly, participants U_7 and U_{14} are located at the end of the Figure 5.13. Simulation of GKAP-MANET can be expressed as follows:

(i) Secure Cluster Head Selection Function: In the execution of the function, each participant $U_i \in \mathcal{U}$ broadcasts list of adjacent participants in its neighborhood as $L_i = N_{U_i}$. Communications and computation costs of participants are only differ with respect to the number of participants of U_i's neighborhood. Since U_3, U_{10} and U_{17} are cluster heads, they have the maximum number of adjacent participants in their neighborhood. Therefore, their communications and computation costs are greater than the other participants as shown in the Figure 5.13.

(ii) Computation of Cluster Keys: As shown in given example network, participants U_7 and U_{14} are categorized as non-clustered participants. Therefore, they are not involve into cluster key computations. Since clusters C_1, C_2 and C_3 have equal number of participants, communications and computations costs of participants in groups are the same, which is 885 ms. for each participant.

(iii) Merge Clusters: After the cluster keys are computed, merging clusters operation is realized. As defined in GKAP-MANET, group merging operation is realized according to ID of the participant U_i. U_6, U_{13} and U_{20} execute GKAP-MANET from Temporary Public Key Distribution Function to Cluster Key Computation Function since they are the participants with the largest ID in their clusters. Moreover, U_5, U_{12} and U_{19} executes GKAP-MANET from Temporary Public Key Verification & Secret Key Distribution Function to Cluster Key Computation Function. The rest of the participants only execute verification functions of the protocol. Therefore, The communications and computation costs of participants can be ordered as follows $U_r > U_s > U_t$, where $r \in \{6, 13, 20\}$, $s \in \{5, 12, 19\}$ and $t \in \{1, 2, 3, 4, 8, 9, 10, 11, 15, 16, 17, 18\}$. Commu-

nications and computation costs of participants are as follows:

$$\mathcal{C}_{comm}(U_r) + \mathcal{C}_{comp}(U_r) = 1372ms \quad (5.6)$$

$$\mathcal{C}_{comm}(U_s) + \mathcal{C}_{comp}(U_s) = 485ms \quad (5.7)$$

$$\mathcal{C}_{comm}(U_t) + \mathcal{C}_{comp}(U_t) = 4655ms \quad (5.8)$$

(iv) Non-clustered Participants Join: To provide connected network, GKAP-MANET uses join operation for non-clustered participants. In our case, since U_7 and U_{14} are not neighbor of any cluster head, they are not participate in any cluster at the beginning. After the merge cluster operation is completed, they join the group by join operation of GKAP-MANET. Since they joins the group after the group key is computed, they are located after U_{20} for simplicity. Moreover, U_{20}, U_7 and U_14 executes GKAP-MANET from Temporary Public Key Distribution Function to Cluster Key Computation Function for updating the group key. Therefore, their communications and computation costs is $1495ms$. In addition, participant U_{19} executes GKAP-MANET from Temporary Public Key Verification & Secret Key Distribution Function to Cluster Key Computation Function and $\mathcal{C}_{comm}(U_{19}) + \mathcal{C}_{comp}(U_{19}) = 175ms$. The total communications and computation costs for the rest of the participants are $121ms$.

(v.) **Participant Leave:** GKAP-MANET is specifically designed by considering the mobile device. Because mobile device can be inactive due to communications problems, mobility or early deplation of batteries, an auxiliary operation is provided for handling participant leaves. In the example, since participant U_{19} leaves the group, participant U_{18} executes GKAP-MANET from Temporary Public Key Distribution Function to Cluster Key Computation Function and participant U_{17} executes GKAP-MANET from Temporary Public Key Verification & Secret Key Distribution Function to Cluster Key Computation Function for updating the group key. Therefore, the communications and computations cost of participant U_{18} is greater than other participants in the group. In addition, participant U_{17} is in the second greatest communications and computation costs. Since rest of the participants only verifies the broadcast messages of U_{17} and U_{18}, they have the same communications and computations costs.

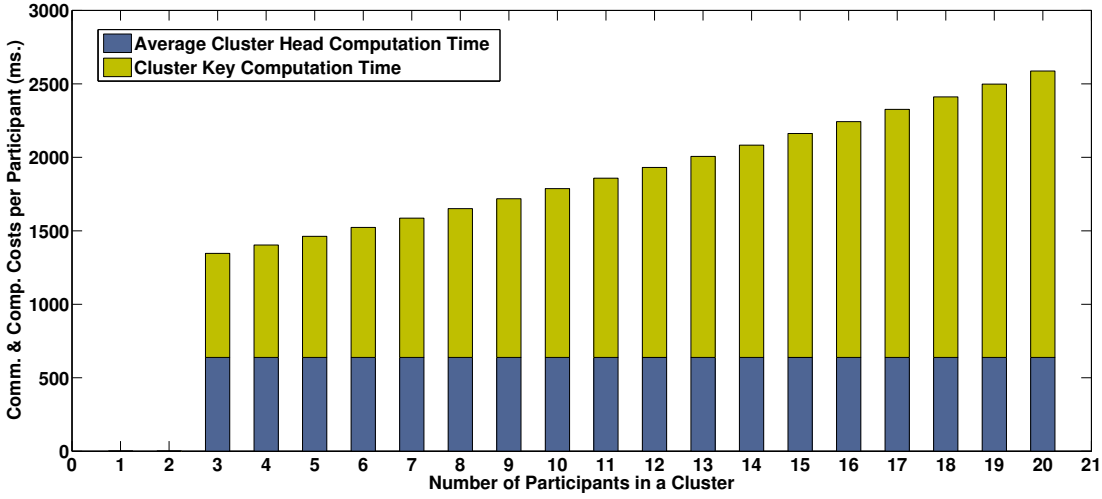

Figure 5.14. Simulation of Communications and Computation Costs per Participant for GKAP-MANET Cluster Head Selection and Cluster Key Computation.

Furthermore, simulation of cluster head selection and cluster key computation for clusters with different number of participants is given in Figure 5.15. As shown in the results, communications and computation costs increase when the number of participant in the cluster is increases. Therefore, in order to compute cluster key in a considerable time, for instance up to five seconds, it is more suitable to execute GKAP-MANET for small clusters up to 50 participants. On the other hand, when we compare the results in Figure 4 with the results in Figure 3, dynamic group operations such as merge, join and leave group key can be more efficiently computed by using formerly computed cluster keys.

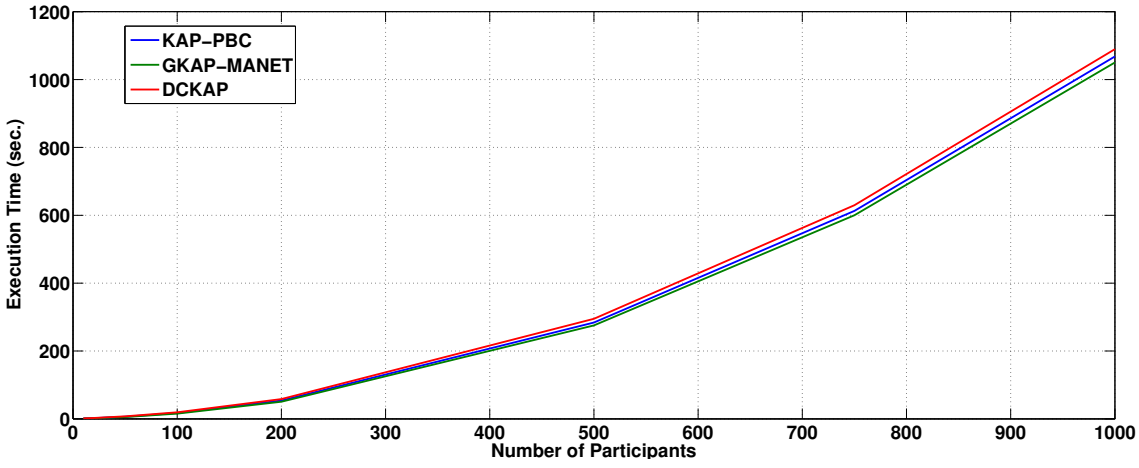

Figure 5.15. Comparisons for Group Key Computation for DCKAP, KAP-PBC and GKAP-MANET

As a final remark, we compare execution time of group key computation for DCKAP, KAP-PBC and GKAP-MANET with respect to the different number of participants in the group. Simulation results show that GKAP-MANET has the best performance in terms of group key computation. However, it is also shown that group key agreement protocols do not a reasonable results for large groups more than 200 participants.

5.7. Discussions

Group key agreement protocols are one of the best candidates for providing secure communications among participants in dynamic group. However, there is a number of problems related to the use of group key agreement protocols in MANETs, such as adaptation in cluster-based communications of MANETs, securely selecting the cluster head for inter-cluster communications and reducing costs of communications and computations during the computation of group key.

In this chapter, we proposed a group key agreement protocol for MANETs. The GKAP-MANET is an improved version of the protocol in [14]. One of these improvements is to provide security against the attacks given in [18]. Thus, the GKAP-MANET provides forward confidentiality and backward confidentiality properties. Moreover, we add merge operation to increase the adaptability of GKAP-MANET cluster-based communications of MANETs. Furthermore, a novel and secure cluster head selection mechanism has been proposed. Analyses show that GKAP-MANET has better performance results than the existing ones in terms of the communications and the computation costs. For the communications cost analysis, the length of the messages exchanged by each participant during the execution of key computation has been considered. For the computation cost analysis, only the modular exponentiation operations of key computation steps have been considered. In short, the performance of GKAP-MANET is independent of the number of participants in the group for both communications cost and the computation cost. Finally, in order to show the applicability of the proposed protocol, we present a set of simulations for an example GKAP-MANET application scenario. As shown in the simulation results, cluster key computation in GKAP-MANET is more suitable for clusters up to 50 participants. GKAP-MANET provides group key for larger groups than 50 participants by using dynamic group operations.

6. CONCLUSION

The pervasive usage of the Internet has made secure group communications a significant issue for several communication applications with dynamic group capabilities. Dynamic group key agreement protocols are one of the best candidates to provide solution for this issue. In this thesis, we have presented our proposals for new protocols research on formal modeling and analysis of group key agreement protocols for the applications on conference communications, secure file sharing systems and communications in MANETs.

6.1. Contributions of the Thesis

First, we have proposed an improved conference-key agreement protocol, called Dynamic Conference-Key Agreement Protocol (DCKAP) for efficient fault correction in conference communications. DCKAP uses a modified form of Tseng's protocol as Initial Conference Key Agreement Protocol (ICKAP) and has new Auxiliary Conference Key Agreement (ACKA) operations to provide dynamic group management. We have analyzed the security of DCKAP and also presented its performance. Our security analysis has shown that DCKAP withstands the known passive and active attacks for conference-key agreement protocols such as eavesdropping and impersonation. In addition, we have also showed that DCKAP preserves correctness, fault-tolerance and forward secrecy properties. Moreover, DCKAP provides better performance results for fault correction than existing conference key agreement protocols. ACKA operations of DCKAP has the linear time computation complexity for conference key update.

To investigate the additional application areas of DCKAP, we have presented a comparative scalability analysis for DCKAP and other dynamic group key agreement protocols via simulations. Simulation results showed that DCKAP can also be used as an efficient participant revocation mechanism in secure file sharing systems. We have proposed a novel file sharing system, namely Three Tier-Secure File Sharing System (TT-SFSS), based on DCKAP to provide efficient participant revocation. In addition to the efficient participant revocation, TT-SFSS provides basic security properties such as escrow-freeness, forward se-

crecy, fine-grained access control and file privacy. Our simulation results have shown that TT-SFSS provides better performance for participant revocation than re-encryption by using ciphertext-policy attribute-based encryption.

Next, we have proposed a new group key agreement protocol, namely Key Agreement Protocol with Partial Backward Confidentiality (KAP-PBC) to overcome the problems of using group key agreement protocols in file sharing systems with lack of privacy, violation of availability and dependency for key escrow in Chapter 4. KAP-PBC provides basic security properties of group key agreement protocols. The security analysis of KAP-PBC against impersonation, eavesdropping and replay attacks are given. In addition, we have presented the performance analysis of KAP-PBC regarding the transmission cost and the computation cost. Complexity analysis of KAP-PBC also shows that dynamic group operations provide efficient performance regarding the normal execution of the protocol.

To be able to use KAP-PBC in file sharing system, we have proposed Private File Sharing System, called PFSS. We have simulated PFSS regarding KAP-PBC service and group update service. Simulation results show that PFSS is applicable for groups up to reasonable number of participants around one hundred participants. Moreover, we have compared the security of PFSS with the existing file sharing systems. According to the comparisons, PFSS is the only system that provides all of the important security properties that we have been addressed.

Finally, we have proposed a group key agreement protocol for MANETs, namely Group Key Agreement Protocols for Mobile Ad hoc Networks (GKAP-MANET). GKAP-MANET is an improved version of the protocol in [14]. One of these improvements is to provide security against the attacks given in [18]. Thus, the GKAP-MANET provides forward confidentiality and backward confidentiality properties. Moreover, we have added merge operation to increase the adaptability of GKAP-MANET cluster-based communications of MANETs. GKAP-MANET contains a novel and secure cluster head selection mechanism. Analyses have shown that GKAP-MANET has better performance results than the existing ones in terms of the communications and the computation costs. For the communications cost analysis, the length of the messages exchanged by each participant during the execution

of key computation has been considered. For the computation cost analysis, only the modular exponentiation operations of key computation steps have been considered. In short, the performance of GKAP-MANET is independent of the number of participants in the group for both communications cost and the computation cost. Finally, in order to show the applicability of the proposed protocol, we have presented a set of simulations for an example GKAP-MANET application scenario. As shown in the simulation results, cluster key computation in GKAP-MANET is more suitable for clusters up to 50 participants. Also, by using dynamic group operations, GKAP-MANET provides group key for larger groups than 50 participants.

6.2. Future Directions

Future directions to improve the research presented in this thesis are as follows:

- In this thesis, we have proposed three different applications of group key agreement protocols for conference communications, secure file sharing systems and communications in MANETs. Group key agreement protocols are one of the best candidate to be used in securing group communication for decentralized and dynamic networks. Therefore, applications of group key agreement protocols can be extended for vehicular ad hoc networks, for example, to take into account their special topology, wireless sensor networks and Internet of Things.
- Group key agreement protocols presented in this thesis are only designed to be used in authenticated networks, where we assume that some authentication information is stored or easily accessible by other participants in the same network. Techniques handling large number of participants in large networks need to be explored for group authentication.
- We have only presented conceptual designs for the use of group key agreement protocols in specific to secure file sharing systems and we have simulated the proposed file system for specific cases. Therefore, implementation of secure cloud-based file sharing system, which is able to provide service for mobile users in a real-time environment can be another future direction.

- Performance analysis of group key agreement protocols and file sharing systems are based on asymptotical analysis and simulations. More formal performance analysis for protocols and systems can be represented such mathematically modeling the join and leave operations of TT-SFSS or PFSS.

REFERENCES

1. Waters, B., "Ciphertext-Policy Attribute-Based Encryption: An Expressive, Efficient, and Provably Secure Realization", *Public Key Cryptography – PKC 2011: 14th International Conference on Practice and Theory in Public Key Cryptography, Taormina, Italy.*, pp. 53–70, 2011.

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

3. Ingemarsson, I., D. T. Tang and C. K. Wong, "A Conference Key Distribution System", *IEEE Transactions on Information Theory*, Vol. 28, pp. 714–719, 1982.

4. Li, C.-H. and J. Pieprzyk, "Conference Key Agreement from Secret Sharing", *ACISP'99: Information Security and Privacy: 4th Australasian Conference*, pp. 64–76, 1999.

5. Chung, Y.-F., "The design of authentication key protocol in certificate-free public key cryptosystem", *Security and Communication Networks*, Vol. 7, pp. 2125–2133, 2013.

6. Tzeng, W.-G., "A Secure Fault-Tolerant Conference-Key Agreement Protocol", *IEEE Transactions on Computers*, Vol. 51, pp. 373–379, 2002.

7. Burmester, M. and Y. Desmedt, "A Secure and Efficient Conference Key Distribution System (Extended Abstract)", *Advances in Cryptology — EUROCRYPT'94: Workshop on the Theory and Application of Cryptographic Techniques*, pp. 275–286, 1994.

8. Amin, R., S. H. Islam, G. Biswas, M. K. Khan, L. Leng and N. Kumar, "Design of an anonymity-preserving three-factor authenticated key exchange protocol for wireless sensor networks", *Computer Networks*, Vol. 101, pp. 42 – 62, 2016.

9. Huang, K.-H., Y.-F. Chung, H.-H. Lee, F. Lai and T.-S. Chen, "A Conference Key

Agreement Protocol with Fault-Tolerant Capability", *Computer Standards and Interfaces*, Vol. 31, pp. 401–405, 2009.

10. Zhao, J., D. Gu and Y. Li, "An Efficient Fault-tolerant Group Key Agreement Protocol", *Comput. Commun.*, Vol. 33, No. 7, pp. 890–895, 2010.

11. Shi, T., Y. Guo and J. Ma, "A Fault-Tolerant and Secure Multi-Conference-Key Agreement Protocol", *International Conference on Communications Circuits and Systems*, pp. 18–21, 2004.

12. Cheng, J.-C. and C.-S. Laih, "Conference key agreement protocol with non-interactive fault-tolerance over broadcast network", *International Journal of Information Security*, Vol. 8, No. 1, pp. 37–48, 2009.

13. Diffie, W., P. C. van Oorschot and M. J. Wiener, "Authentication and Authenticated Key Exchanges", *Design, Codes and Cryptography*, Vol. 2, pp. 107–125, 1992.

14. Tseng, Y.-M., "An Improved Conference-Key Agreement Protocol with Forward Secrecy", *Informatica*, Vol. 16, pp. 275–284, 2005.

15. Ermis, O., S. Bahtiyar, E. Anarim and M. U. Çaglayan, "An improved fault-tolerant conference-key agreement protocol with forward secrecy", *SIN13: The 6th International Conference on Security of Information and Networks Aksaray, Turkey*, pp. 306–310, 2013.

16. Ermis, O., S. Bahtiyar, E. Anarim and M. U. Caglayan, "An improved conference-key agreement protocol for dynamic groups with efficient fault correction", *Security and Communication Networks*, Vol. 8, pp. 1347–1359, 2015.

17. Tseng, Y.-M., "A communication-efficient and fault-tolerant conference-key agreement protocol with forward secrecy", *The Journal of Systems and Software*, Vol. 80, pp. 1091–1101, 2007.

18. Lee, S., J. Kim and S. J. Hong, “Security weakness of Tseng’s fault-tolerant conference key agreement protocol”, *The Journal of Systems and Software*, Vol. 82, pp. 1163–1167, 2009.

19. Tzeng, W.-G., “A Practical and Secure Fault-Tolerant Conference-Key Agreement Protocol”, *Public Key Cryptography: Third International Workshop on Practice and Theory in Public Key Cryptosystems, PKC 2000, Melbourne, Victoria, Australia*, 2000.

20. D.E. Denning, D. B., “A Taxonomy for Key Escrow Encryption Systems”, *Communcations of the ACM*, Vol. 39, No. 3, pp. 34–40, March 1996.

21. Fu, A., G. Zhang and Z. Zhu, “A Secure and Efficient Fault-Tolerant Group Key Agreement Protocol”, *12th IEEE International Conference on Trust, Security and Privacy in Computing and Communications*, pp. 310–314, 2013.

22. Lei, X., X. Liao and Y. Xiong, “Group Key Agreement Protocol for MANETs Based on HSK Scheme”, *CoRR*, Vol. abs/1312.3420, 2013.

23. Ermis, O., S. Bahtiyar, E. Anarim and M. U. Caglayan, “A Comparative Study on the Scalability of Dynamic Group Key Agreement Protocols”, *ARES: International Conference on Availability, Reliability and Security*, 2017.

24. Ermis, O., S. Bahtiyar, E. Anarim and M. U. Caglayan, “A Key Agreement Protocol with Partial Backward Confidentiality”, *Submitted to Elsevier Computer Networks Journal (Major Revision Response has been sent)*, 2017.

25. Ermis, O., S. Bahtiyar, E. Anarim and M. U. Caglayan, “Open problems for group-key agreement protocols on Vehicular Ad-hoc Networks”, *ICCVE: International Conference on Connected Vehicles and Expo*, pp. 828–831, 2013.

26. Ermis, O., E. Anarim and M. U. Caglayan, “Yeni Hata Toleransli Konferans Anahtari Anlasma Protokolu”, *Akademik Bilisim, Antalya*, 2013.

27. Ermis, O., S. Bahtiyar, E. Anarim and M. U. Caglayan, “Grup Anahtari Olusturma Protokolleri ve Uygulamalari”, *Akademik Bilisim, Mersin*, 2014.

28. Catakli, T., O. Ermis, C. Tunca, S. Isik, C. Ersoy and M. U. Caglayan, “Kablosuz Algilayici Aglarinda Grup Anahtari Degisim Protokollerinin Enerji Basarimi Degerlendirmeleri”, *Akademik Bilisim, Aydin*, 2016.

29. Cantali, G., O. Ermis, G. Gur, F. Alagoz and M. U. Caglayan, “Lightweight context-aware security system for wireless Internet access”, *CNS: IEEE Conference on Communications and Network Security*, pp. 765–766, 2015.

30. Cantali, G., O. Ermis, G. Gur, F. Alagoz and M. U. Caglayan, “Kablosuz Internet Erisimi icin Hafif Siklet Baglam Bilincli Ag Guvenlik Sistemi”, *Akademik Bilisim, Aydin*, 2016.

31. Bahtiyar, Ş., O. Ermiş and M. U. Çağlayan, “Adaptive Trust Scenarios for Mobile Security”, *MobiWIS: Mobile Web and Intelligent Information Systems: 13th International Conference, Vienna, Austria*, pp. 137–148, 2016.

32. Erdem, M. D., O. Ermis, C. Tunca, S. Isik, C. Ersoy and M. U. Caglayan, “Kablosuz Algilayici Aglarinda Grup Anahtari Yonetim Protokollerinin Basarim Degerlendirmesi”, *Akademik Bilisim, Eskisehir*, 2015.

33. Ermis, O., S. Bahtiyar, E. Anarim and M. U. Caglayan, “A Secure and Efficient Group Key Agreement Approach for Mobile Ad Hoc Networks”, *Submitted to Elsevier Ad Hoc Networks Journal (Under Review)*, 2016.

34. Bahtiyar, S., O. Ermis and M. U. Caglayan, “A Framework for Trust Assessment of Security Systems on Flexible Networks”, *FiCloud: Submitted to The 5th International Conference on Future Internet of Things and Cloud*, 2017.

35. Tzeng, W.-G. and Z.-J. Tzeng, “Round-Efficient Conference Key Agreement Protocols with Provable Security”, *ASIACRYPT’ 00: Proceedings of the 6th International*

Conference on the Theory and Application of Cryptology and Information Security: Advances in Cryptology, pp. 614–628, 2000.

36. Wu, T.-Y. and Y.-M. Tseng, "Towards ID-Based Authenticated Group Key Exchange Protocol with Identifying Malicious Participants", *Informatica, Lith. Acad. Sci.*, Vol. 23, pp. 315–334, 2012.

37. Trappe, W., Y. Wang and K. J. R. Liu, "Resource-Aware Conference Key Establishment for Heterogeneous Networks", *IEEE/ACM Transactions on Networking*, Vol. 13, pp. 134–146, 2005.

38. Boneh, D., "The Decision Diffie-Hellman Problem", *Proceedings of the Third International Symposium on Algorithmic Number Theory*, 1998.

39. Abramowitz, M., *Handbook of Mathematical Functions, With Formulas, Graphs, and Mathematical Tables,*, Dover Publications, Incorporated, 1974.

40. Goldreich, O., *Foundations of Cryptography: Volume 1*, Cambridge University Press, New York, NY, USA, 2006.

41. Stinson, D., *Cryptography: Theory and Practice,Second Edition*, CRC/C&H, 2nd edn., 2002.

42. Rivest, R. L., A. Shamir and L. Adleman, "A Method for Obtaining Digital Signatures and Public-key Cryptosystems", *Communications of the ACM*, Vol. 21, No. 2, pp. 120–126, 1978.

43. ElGamal, T., "A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms", *CRYPTO 84: Advances in Cryptology: Proceedings of International Cryptology Conference*, pp. 10–18, 1985.

44. Maurer, U. M., "Towards the Equivalence of Breaking the Diffie-Hellman Protocol and Computing Discrete Logarithms", pp. 271–281, 1994.

45. Bellare, M. and P. Rogaway, “Random oracles are practical: a paradigm for designing efficient protocols”, *Proceedings of the 1st ACM Conference on Computer and Communications Security*, pp. 62–73, 1993.

46. Dutta, R. and R. Barua, “Overview of Key Agreement Protocols”, *IACR Cryptology ePrint Archive*, Vol. 2005, pp. 289–334, 2005.

47. Katz, J. and M. Yung, “Scalable Protocols for Authenticated Group Key Exchange”, *Journal of Cryptology*, Vol. 20, pp. 85–113, 2007.

48. Steiner, M., G. Tsudik and M. Waidner, “Key Agreement in Dynamic Peer Groups”, *IEEE Transactions on Parallel and Distributed Systems*, Vol. 11, pp. 769–780, 2000.

49. Horng, G., “An Efficient and Secure Protocol for Multi-Party Key Establishment”, *The Computer Journal*, Vol. 44, pp. 463–470, 2001.

50. Ateniese, G., M. Steiner and G. Tsudik, “New multiparty authentication services and key agreement protocols”, *IEEE Journal on Selected Areas in Communications*, Vol. 18, No. 4, pp. 628–639, 2000.

51. Steiner, M., G. Tsudik and M. Waidner, “CLIQUES: a new approach to group key agreement”, *18th International Conference on Distributed Computing Systems Proceedings*, pp. 380–387, 1998.

52. Bresson, E., O. Chevassut and D. Pointcheval, “Provably Secure Authenticated Group Diffie-Hellman Key Exchange”, *ACM Trans. Information and System Security*, Vol. 10, No. 3, 2007.

53. Bresson, E., O. Chevassut and D. Pointcheval, “Dynamic Group Diffie-Hellman Key Exchange under Standard Assumptions”, *Advances in Cryptology — EUROCRYPT: International Conference on the Theory and Applications of Cryptographic Techniques*, pp. 321–336, 2002.

54. "SEC 1. Standards for Efficient Cryptography Group: Elliptic Curve Cryptography", Certicom Research, 2009.

55. "ANSI X9.63-2011 (R2017), Public-Key Cryptography for the Financial Services Industry: Key Agreement and Key Transport Using Elliptic Curve Cryptography", American National Standards Institute, 2011.

56. B. Zan, M. G., and and F. Hu, "Key Agreement Algorithms for Vehicular Communication Networks Based on Reciprocity and Diversity Theorems", *IEEE Transactions on Vehicular Technology*, Vol. 62, No. 8, pp. pp. 4020–4027.

57. Almeida, J., S. Shintre, M. Boban and J. Barros, "Probabilistic Key Distribution in Vehicular Networks with Infrastructure Support", *IEEE Global Communications Conference, GLOBECOM*, pp. 973–978, 2012.

58. Erritali, M., O. Reda and B. Ouahidi, "A Beaconing Approach with Key Exchange in Vehicular Ad Hoca Networks", *International Journal of Distributed and Parallel Systems*, Vol. 3, No. 6, pp. 9–13, 2012.

59. D. Kim, J. C. and S. Jung, "Mutual Identification and Key Exchange Scheme in Secure VANETs based on Group Signature", *Proceedings of the 7th IEEE Conference on Consumer Communications and Networking Conference*, pp. 1–2, 2010.

60. Hegland, A. M., E. Winjum, S. F. Mjolsnes, C. Rong, O. Kure and P. Spilling, "A survey of key management in ad hoc networks", *IEEE Communications Surveys Tutorials*, Vol. 8, No. 3, pp. 48–66, 2006.

61. Zhao, S., A. Aggarwal, R. Frost and X. Bai, "A Survey of Applications of Identity-Based Cryptography in Mobile Ad-Hoc Networks", *IEEE Communications Surveys and Tutorials*, Vol. 14, No. 2, pp. 380–400, 2012.

62. Huang, D. and D. Medhi, "A Secure Group Key Management Scheme for Hierarchical Mobile Ad Hoc Networks", *Ad Hoc Networks*, Vol. 6, No. 4, pp. 560–577, Jun. 2008.

63. Wang, N.-C. and S.-Z. Fang, “A Hierarchical Key Management Scheme for Secure Group Communications in Mobile Ad Hoc Networks”, *Journal of Systems and Software*, Vol. 80, No. 10, pp. 1667–1677, 2007.

64. Chen, Q., Z. M. Fadlullah, X. Lin and N. Kato, “A clique-based secure admission control scheme for mobile ad hoc networks (MANETs).”, *Journal of Network and Computer Applications*, Vol. 34, No. 6, pp. 1827–1835, 2011.

65. Eschenauer, L. and V. D. Gligor, “A Key-management Scheme for Distributed Sensor Networks”, *Proceedings of the 9th ACM Conference on Computer and Communications Security*, CCS ’02, pp. 41–47, 2002.

66. Rasheed, A. and R. N. Mahapatra, “The Three-Tier Security Scheme in Wireless Sensor Networks with Mobile Sinks”, *IEEE Transactions on Parallel and Distributed Systems*, Vol. 23, No. 5, pp. 958–965, 2012.

67. Zhu, Z. and R. Jiang, “A Secure Anti-Collusion Data Sharing Scheme for Dynamic Groups in the Cloud”, *IEEE Transactions on Parallel and Distributed Systems*, Vol. 27, No. 1, pp. 40–50, 2016.

68. Liu, X., Y. Zhang, B. Wang and J. Yan, “Mona: Secure Multi-Owner Data Sharing for Dynamic Groups in the Cloud”, *Parallel and Distributed Systems, IEEE Transactions on*, Vol. 24, No. 6, pp. 1182–1191, 2013.

69. Liu, Z., J. Li, X. Chen, J. Yang and C. Jia, “TMDS: Thin-Model Data Sharing Scheme Supporting Keyword Search in Cloud Storage”, *Information Security and Privacy: 19th Australasian Conference, ACISP, Wollongong, NSW, Australia*, 2014.

70. Tang, Y., P. P. C. Lee, J. C. S. Lui and R. Perlman, “Secure Overlay Cloud Storage with Access Control and Assured Deletion”, *IEEE Trans. Dependable Secur. Comput.*, Vol. 9, No. 6, pp. 903–916, 2012.

71. Lim, H. W. and G. Yang, “Authenticated Key Exchange Protocols for Parallel Network

File Systems", *IEEE Trans. Parallel Distrib. Syst.*, Vol. 27, No. 1, pp. 92–105, 2016.

72. Emery, S., "Kerberos Version 5 Generic Security Service Application Program Interface (GSS-API) Channel Binding Hash Agility", Internet Eng. Task Force (IETF), 2012.

73. Sandhu, R. S., E. J. Coyne, H. L. Feinstein and C. E. Youman, "Role-Based Access Control Models", *IEEE Computer*, Vol. 29, No. 2, pp. 38–47, 1996.

74. Li, M., S. Yu, Y. Zheng, K. Ren and W. Lou, "Scalable and Secure Sharing of Personal Health Records in Cloud Computing Using Attribute-Based Encryption", *Parallel and Distributed Systems, IEEE Transactions on*, Vol. 24, No. 1, pp. 131–143, 2013.

75. Liu, Q., G. Wang and J. Wu, "Time-based proxy re-encryption scheme for secure data sharing in a cloud environment", *Information Sciences*, Vol. 258, pp. 355 – 370, 2014.

76. Sahai, A. and B. Waters, "Fuzzy Identity-based Encryption", *EUROCRYPT'05: Proceedings of the 24th Annual International Conference on Theory and Applications of Cryptographic Techniques*, pp. 457–473, 2005.

77. Goyal, V., O. Pandey, A. Sahai and B. Waters, "Attribute-based Encryption for Fine-grained Access Control of Encrypted Data", *CCS '06: Proceedings of the 13th ACM Conference on Computer and Communications Security*, pp. 89–98, 2006.

78. Bethencourt, J., A. Sahai and B. Waters, "Ciphertext-Policy Attribute-Based Encryption", *SP '07: Proceedings of the IEEE Symposium on Security and Privacy*, pp. 321–334, 2007.

79. Wang, G., Q. Liu, J. Wu and M. Guo, "Hierarchical Attribute-based Encryption and Scalable User Revocation for Sharing Data in Cloud Servers", *Computers and Security*, Vol. 30, No. 5, pp. 320–331, 2011.

80. Arya, P. K., K. Selvamani and S. Kanimozhi, "An authentication approach for data

sharing in cloud environment for dynamic group", *ICICT: International Conference on Issues and Challenges in Intelligent Computing Techniques*, pp. 262–267, 2014.

81. Blaze, M., G. Bleumer and M. Strauss, "Divertible protocols and atomic proxy cryptography", *Advances in Cryptology — EUROCRYPT'98: International Conference on the Theory and Application of Cryptographic Techniques Espoo, Finland*, pp. 127–144, 1998.

82. Canetti, R. and S. Hohenberger, "Chosen-ciphertext Secure Proxy Re-encryption", *CCS '07: Proceedings of the 14th ACM Conference on Computer and Communications Security*, pp. 185–194, 2007.

83. Green, M. and G. Ateniese, "Identity-Based Proxy Re-encryption", *Applied Cryptography and Network Security: 5th International Conference, ACNS'07, Zhuhai, China*, pp. 288–306, 2007.

84. Ateniese, G., K. Fu, M. Green and S. Hohenberger, "Improved Proxy Re-encryption Schemes with Applications to Secure Distributed Storage", *ACM Transactions on Information and Systems Security*, Vol. 9, No. 1, pp. 1–30, 2006.

85. Khurana, H., A. Slagell and R. Bonilla, "SELS: A Secure e-Mail List Service", *Proceedings of the ACM Symposium on Applied Computing*, pp. 306–313, 2005.

86. Shao, J. and Z. Cao, "CCA-Secure Proxy Re-encryption without Pairings", *Public Key Cryptography – PKC: 12th International Conference on Practice and Theory in Public Key Cryptography, Irvine, CA, USA*, pp. 357–376, 2009.

87. Xiong, H., X. Zhang, D. Yao, X. Wu and Y. Wen, "Towards End-to-end Secure Content Storage and Delivery with Public Cloud", *Proceedings of the Second ACM Conference on Data and Application Security and Privacy*, pp. 257–266, 2012.

88. Housley, R., W. Polk, W. Ford and D. Solo, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 5280 (Proposed

Standard), 2002.

89. Backes, M., C. Cachin and A. Oprea, "Lazy Revocation in Cryptographic File Systems", *3rd International IEEE Security in Storage Workshop, San Francisco, California, USA*, pp. 1–11, 2005.

90. Yu, S., C. Wang, K. Ren and W. Lou, "Achieving Secure, Scalable, and Fine-grained Data Access Control in Cloud Computing", *INFOCOM'10: Proceedings of the 29th Conference on Information Communications*, pp. 534–542, 2010.

91. Zhou, Z. and D. Huang, "Efficient and Secure Data Storage Operations for Mobile Cloud Computing", *CNSM '12: Proceedings of the 8th International Conference on Network and Service Management*, pp. 37–45, 2013.

92. Steiner, M., G. Tsudik and M. Waidner, "A fault-tolerant and secure multi-conference-key agreement protocol", *CCS '96 Proceedings of the 3rd ACM conference on Computer and communications security*, pp. 890–895, 1996.

93. Foss, J. A., "An Efficient Secure Authenticated Group Key Exchange Algorithm for Large and Dynamic Groups", *NISSC: Proc. 23rd Nat'l Information Systems Security Conference*, pp. 254–266, 2000.

94. Boyd, C. and J. M. G. Nieto, "Round-Optimal Contributory Conference Key Agreement", *Public Key Cryptography — PKC: 6th International Workshop on Practice and Theory in Public Key Cryptography*, pp. 161–174, 2003.

95. Chang, C.-C., H.-C. Tsai and P.-Y. Chang, "A Collaborative Conference Key Agreement Scheme by Using an Intermediary Node", *Proceedings of the 2007 International Conference on Convergence Information Technology*, pp. 54–59, 2007.

96. Wu, Q., Y. Mu, W. Susilo, B. Qin and J. Domingo-Ferrer, "Asymmetric Group Key Agreement", *Advances in Cryptology - EUROCRYPT, 28th Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pp. 153–170,

2009.

97. Wang, Z., “Improvement on the fault-tolerant group key agreement protocol of Zhao et al”, *Security and Communication Networks*, Vol. 9, pp. 166–170, 2012.

98. Cheng, Z.-Y., Y. Liu, C.-C. Chang and C. Guo, “A fault-tolerant group key agreement protocol exploiting dynamic setting”, *International Journal of Communication Systems*, Vol. 26, pp. 259–275, 2013.

99. Pointcheval, D. and J. Stern, “Security Proofs for Signature Schemes”, *Advances in Cryptology — EUROCRYPT: International Conference on the Theory and Application of Cryptographic Techniques Saragossa, Spain*, pp. 387–398, 1996.

100. Pointcheval, D. and J. Stern, “Security Arguments for Digital Signatures and Blind Signatures”, *Journal of Cryptography*, Vol. 13, pp. 361–396, 2000.

101. Li, M., X. Xu, C. Guo and X. Tan, “AD-ASGKA authenticated dynamic protocols for asymmetric group key agreement”, *Security and Communication Networks*, pp. 1340–1352, 2016.

102. Zhang, L., Q. Wu, J. Domingo-Ferrer, B. Qin and Z. Dong, “Round-Efficient and Sender-Unrestricted Dynamic Group Key Agreement Protocol for Secure Group Communications”, *Information Forensics and Security, IEEE Transactions on*, Vol. 10, No. 11, pp. 2352–2364, 2015.

103. Akinyele, J. A., C. Garman, I. Miers, M. W. Pagano, M. Rushanan, M. Green and A. D. Rubin, “Charm: a framework for rapidly prototyping cryptosystems”, *Journal of Cryptographic Engineering*, Vol. 3, No. 2, pp. 111–128, 2013.

104. Pussewalage, H. S. G. and V. Oleshchuk, “A Patient-Centric Attribute Based Access Control Scheme for Secure Sharing of Personal Health Records Using Cloud Computing”, *IEEE 2nd International Conference on Collaboration and Internet Computing (CIC)*, pp. 46–53, 2016.

105. Wang, S., J. Zhou, J. K. Liu, J. Yu, J. Chen and W. Xie, "An Efficient File Hierarchy Attribute-Based Encryption Scheme in Cloud Computing", *IEEE Transactions on Information Forensics and Security*, Vol. 11, No. 6, pp. 1265–1277, 2016.

106. Yu, S., C. Wang, K. Ren and W. Lou, "Attribute Based Data Sharing with Attribute Revocation", *ASIACCS '10: Proceedings of the 5th ACM Symposium on Information, Computer and Communications Security*, pp. 261–270, 2010.

107. Choi, C. H., "Adoptation of Weil Pairing IBE for Secure File Sharing", *GlobeNet: The Fifth International Conference on Advances in Databases, Knowledge, and Data Applications*, pp. 59–65, 2013.

108. Malarvizhi, M., J. Sujana and T. Revathi, "Secure File Sharing Using Cryptographic Techniques in Cloud", *Green Computing Communication and Electrical Engineering (ICGCCEE), International Conference on*, pp. 1–6, 2014.

109. Solapurkar, P., "Secure sharing of personal health records on cloud using key-aggregate cryptosystem", *ICIP: International Conference on Information Processing*, pp. 278–283, 2015.

110. Zhao, X., F. Zhang and H. Tian, "Dynamic asymmetric group key Agreement for ad hoc networks", *Ad Hoc Networks*, Vol. 9, pp. 928–939, 2011.

111. Amin, R., N. Kumar, G. Biswas, R. Iqbal and V. Chang, "A light weight authentication protocol for IoT-enabled devices in distributed Cloud Computing environment", *Future Generation Computer Systems*, pp. –, 2016.

112. Amin, R., S. H. Islam, G. Biswas, M. K. Khan and N. Kumar, "A robust and anonymous patient monitoring system using wireless medical sensor networks", *Future Generation Computer Systems*, pp. –, 2016.

113. Sureshkumar, V., R. Anitha, N. Rajamanickam and R. Amin, "A lightweight two-gateway based payment protocol ensuring accountability and unlinkable anonymity

with dynamic identity", *Computers & Electrical Engineering*, Vol. 57, pp. 223 – 240, 2017.

114. Kim, Y., A. Perrig and G. Tsudik, "Tree-Based Group Key Agreement", *ACM Transactions on Information and System Security*, Vol. 7, pp. 60–96, 2004.

115. Hao, G., N. V. Vinodchandran, B. Ramamurthy and X. Zou, "A balanced key tree approach for dynamic secure group communication", *Proceedings of the 14th International Conference On Computer Communications and Networks, ICCCN, San Diego, California, USA*, pp. 345–350, 2005.

116. Xiaozhuo, G., X. Taizhong, Z. Weihua and W. Yongming, "A Pairing-Free Certificateless Authenticated Group Key Agreement Protocol", *High Performance Computing and Communications, IEEE 6th Intl Symp on Cyberspace Safety and Security*, pp. 510–513, 2014.

117. Mortazavi, S., A. Pour and T. Kato, "An efficient distributed group key management using hierarchical approach with Diffie-Hellman and Symmetric Algorithm: DHSA", *Computer Networks and Distributed Systems (CNDS), 2011 International Symposium on*, pp. 49–54, 2011.

118. Zhang, L., Q. Wu, B. Qin, J. Domingo-Ferrer and U. Gonzalez-Nicolas, "Asymmetric group key agreement protocol for open networks and its application to broadcast encryption", *Computer Networks*, Vol. 55, No. 15, pp. 3246 – 3255, 2011.

119. Xia, Q., J. Ni, A. J. B. A. Kanpogninge and J. C. Gee, "Searchable Public-Key Encryption with Data Sharing in Dynamic Groups for Mobile Cloud Storage", *Journal of Universal Computer Science*, Vol. 21, No. 3, pp. 440–453, 2015.

120. Bellare, M. and A. Palacio, "GQ and Schnorr Identification Schemes: Proofs of Security against Impersonation under Active and Concurrent Attacks", M. Yung (Editor), *Advances in Cryptology — CRYPTO: 22nd Annual International Cryptology Conference Santa Barbara, California, USA*, pp. 162–177, 2002.

121. Morita, H., J. C. N. Schuldt, T. Matsuda, G. Hanaoka and T. Iwata, "On the Security of the Schnorr Signature Scheme and DSA Against Related-Key Attacks", S. Kwon and A. Yun (Editors), *Information Security and Cryptology - ICISC: 18th International Conference, Seoul, South Korea, November 25-27, 2015, Revised Selected Papers*, pp. 20–35, 2016.

122. Polk, T., K. McKay, S. Chokhani, I. T. L. N. I. of Standards and Technology, *Guidelines for the Selection, Configuration, and Use of Transport Layer Security (TLS) Implementations*, NIST special publication, 2014.

123. Dutta, R. and T. Dowling, "Provably Secure Hybrid Key Agreement Protocols in Cluster-based Wireless Ad Hoc Networks", *Ad Hoc Networks*, Vol. 9, No. 5, pp. 767–787, 2011.

124. He, D., "An Efficient Remote User Authentication and Key Agreement Protocol for Mobile Client-server Environment from Pairings", *Ad Hoc Networks*, Vol. 10, No. 6, pp. 1009–1016, 2012.

125. Durahim, A. O. and E. Savas, "A^2-MAKE: An efficient anonymous and accountable mutual authentication and key agreement protocol for WMNs", *Ad Hoc Networks*, Vol. 9, No. 7, pp. 1202–1220, 2011.

126. Farash, M. S., M. Turkanović, S. Kumari and M. Hölbl, "An Efficient User Authentication and Key Agreement Scheme for Heterogeneous Wireless Sensor Network Tailored for the Internet of Things Environment", *Ad Hoc Netw.*, Vol. 36, No. P1, pp. 152–176, 2016.

127. Zhou, L. and C. V. Ravishankar, "Efficient, Authenticated, and Fault-Tolerant Key Agreement for Dynamic Peer Groups", *Networking 2004: Networking Technologies, Services, and Protocols; Performance of Computer and Communication Networks; Mobile and Wireless Communications Third International IFIP-TC6 Networking Conference Athens, Greece*, pp. 759–770, 2004.

128. Zou, X., A. Thukral and B. Ramamurthy, “An Authenticated Key Agreement Protocol for Mobile Ad Hoc Networks”, *Mobile Ad-hoc and Sensor Networks: Second International Conference, MSN, Hong Kong, China*, pp. 509–520, 2006.

129. Bhaskar, R., D. Augot, Ć. Adjih, P. Mühlethaler and S. Boudjit, “AGDH (Asymmetric Group Diffie Hellman) An Efficient and Dynamic Group Key Ageement Protocol for Ad Hoc Networks”, *New Technologies, Mobility and Security*, pp. 633–633, 2007.

130. Sun, H.-M., B.-Z. He, C.-M. Chen, T.-Y. Wu, C.-H. Lin and H. Wang, “A provable authenticated group key agreement protocol for mobile environment.”, *Inf. Sci.*, Vol. 321, pp. 224–237, 2015.

131. Tan, C. H. and J. C. M. Teo, “Energy-efficient ID-based Group Key Agreement Protocols for Wireless Networks”, *Proceedings of the 20th International Conference on Parallel and Distributed Processing*, pp. 356–356, Washington, DC, USA, 2006.

132. Das, P. and V. Gulhane, “Distributed Energy Adaptive Location-Based Cooperative MAC Protocol for Prolonging the Network Lifetime of MANET”, *Proceedings of the Fifth International Conference on Advanced Computing & Communication Technologies*, pp. 298–303, 2015.

133. Halford, T. R., T. A. Courtade and K. M. Chugg, “Energy-efficient, secure group key agreement for ad hoc networks”, *Communications and Network Security (CNS), 2013 IEEE Conference on*, pp. 181–188, 2013.

134. Magliveras, S. S., W. Wei and X. Zou, “Notes on the CRTDH Group Key Agreement Protocol”, *The 28th International Conference on Distributed Computing Systems Workshops*, pp. 406–411, 2008.

135. Dutta, R. and R. Barua, “Dynamic group key agreement in tree-based setting”, *Information Security and Privacy: 10th Australasian Conference, ACISP 2005, Brisbane, Australia*, pp. 101–112, 2005.

136. Perrig, A., “Efficient collaborative key management protocols for secure autonomous group communication”, *In Proceedings of International workshop on cryptographic techniques and electronic commerce*, pp. 192–202, 1999.