SECURE GROUP COMMUNICATION IN GRID

A DISSERTATION

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

INTEGRATED DUAL DEGREE

in

COMPUTER SCIENCE AND ENGINEERING
(With Specialization in Information Technology)

By

SRI KRISHNA CHOWDARY K

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

JUNE, 2012

CANDIDATE'S DECLARATION

I hereby declare that the work being presented in the dissertation work entitled "**Secure Group Communication in Grid**" towards the partial fulfillment of the requirement for the award of the degree of **Integrated Dual Degree in Computer Science and Engineering (with specialization in Information Technology)** and submitted to the **Department of Electronics and Computer Engineering, Indian Institute of Technology Roorkee, India** is an authentic record of my own work carried out during the period from May, 2011 to June, 2012 under the guidance and provision of **Dr. Anjali Sardana, Assistant Professor, Department of Electronics and Computer Engineering, IIT Roorkee and Dr. P Sateesh Kumar, Assistant Professor, Department of Electronics and Computer Engineering, IIT Roorkee.**

I have not submitted the matter embodied in this dissertation work for the award of any other degree and diploma.

Date: June, 2012

Place: Roorkee

[illegible]

(SRI KRISHNA CHOWDARY K)

CERTIFICATE

This is to certify that the declaration made by the candidate above is correct to the best of our knowledge and belief.

Date: June, 2012

Place: Roorkee

Anjali Sardana 14/06/12

Dr. Anjali Sardana
Assistant Professor,
E&CE Department
IIT Roorkee, India

[illegible]

Dr. P Sateesh Kumar
Assistant Professor,
E&CE Department
IIT Roorkee, India

Dr. P. SATEESH KUMAR
Assistant Professor
Deptt. of Electronics & Computer Engg.
Indian Institute of Technology Roorkee
Roorkee-247667 (U.K.) INDIA

ACKNOWLEDGEMENTS

I would like to take this opportunity to extend my heartfelt gratitude to my guides and mentors **Dr. Anjali Sardana, Assistant Professor, Department of Electronics and Computer Engineering, Indian Institute of Technology Roorkee**, and **Dr. P. Sateesh Kumar , Assistant Professor, Department of Electronics and Computer Engineering, Indian Institute of Technology Roorkee**, for their trust in my work, their able guidance, regular source of encouragement and assistance throughout this dissertation work. I would state that the dissertation work would not have been in the present shape without their inspirational support and I consider myself fortunate to have done my dissertation under them.

I also extend my sincere thanks to **Dr. Padam Kumar, Professor and Head of the Department of Electronics and Computer Engineering** for providing facilities for the work.

I would like to thank all my friends especially Rajdeep Barua who supported and encouraged me to finish this work.

Finally, I would like to say that I am indebted to my parents for everything that they have given to me. I thank them for sacrifices they made so that I could grow up in a learning environment. They have always stood by me in everything I have done, providing constant support, encouragement, and love.

SRI KRISHNA CHOWDARY K

ABSTRACT

Grid environment is characterised by groups that enable coordinated resource sharing among heterogeneous users/resources. The membership in grid is large and dynamic. There is a need for a SGC mechanism that is scalable as well as handles membership changes efficiently since security should not compromise the performance of grid. The dissertation focuses on providing a secure, scalable and efficient mechanism for group communication in grid. Some of the mechanisms have been analysed, of which some are found to be not secure enough while being computationally efficient. Solutions have been provided to make them secure. Some others are found to be secure but high computational overhead is involved. While each mechanism has its advantages and disadvantages, few solutions have been proposed by combining the existing techniques which are supposed to improve security while performance is not degraded. The proposed mechanisms have been implemented. Simulations have been run to prove that some of the methods are not secure and also analyse the performance of various schemes. The results of these have been provided that verify the same.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1
INTRODUCTION

The grid problem [1] is to enable coordinated resource sharing and problem solving in dynamic, multi institutional virtual organizations. The resource sharing in grid is, necessarily, highly controlled, with resource providers and consumers defining clearly and carefully just what is shared, who is allowed to share, and the conditions under which sharing occurs. A set of individuals and/or institutions defined by such sharing rules form a virtual organization (VO). Grid computing finds very much application in financial applications, bioinformatics, etc.

1.1. Secure Communication in Grid

The security to enable virtual organization makes grid security different form conventional security. VO resources and users are often located in distinct administrative domains [2]. Different administrative domains may have different mechanisms and credentials. Grid uses the internet and networks for communication and they are not security oriented by design, passive and active attacks can be launched on grid from outside members. So, any communication between the members of the grid, the users and the resources should be confidential. Public key cryptographic algorithms like RSA are used in grid for achieving confidentiality. The sender can encrypt the message using public key of the receiver and the receiver can decode using his private key [3]. The public key infrastructure is responsible for any entity to get the public key of any other entity as shown in figure 1.1. In addition to secure communication, grid security needs to have authentication, access control, integrity, privacy, nonrepudiation and more over single sign on, policy exchange, manageability, interoperability and exportability [4]. Since, the main goal of grid computing is performance improvement; the performance of a grid computing should not be affected much due to the introduction of security [5]. The entities in grid may have very different functionalities system administrators, sensors, resources etc. Most of the entities are such that they have dedicated roles to perform in the grid and communicate with only those entities that are required in the task that it is

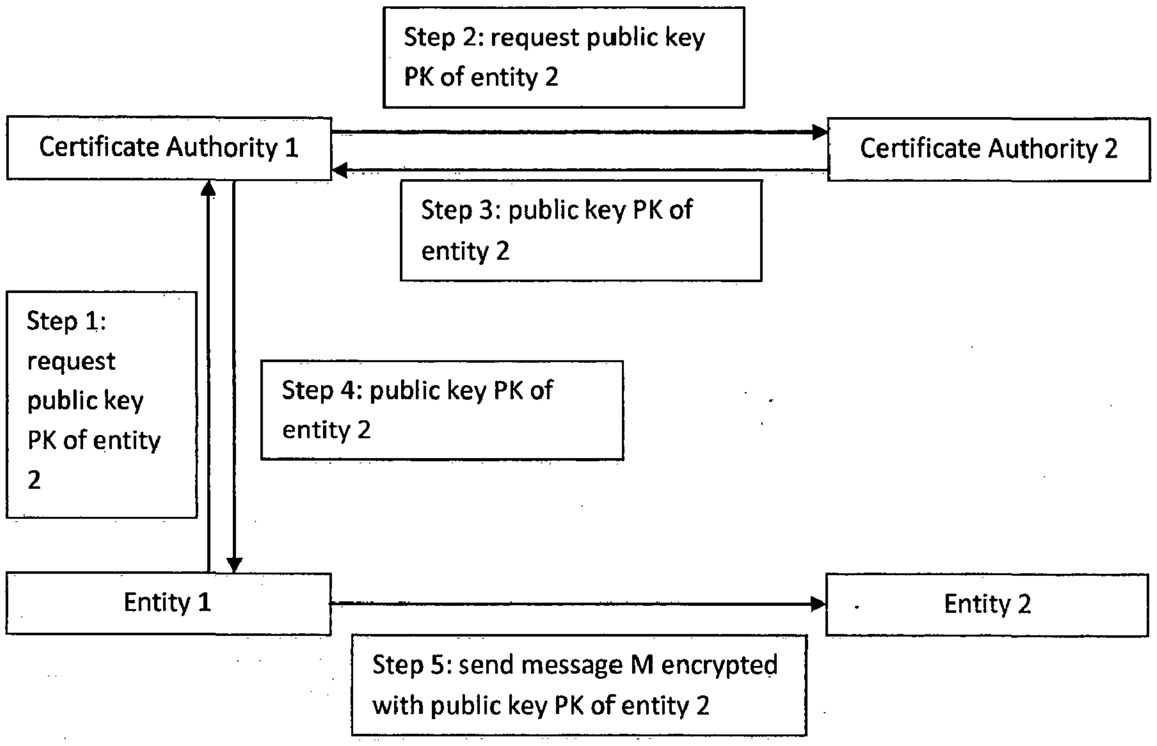

Step 2: request public key PK of entity 2
Certificate Authority 1
Certificate Authority 2
Step 3: public key PK of entity 2
Step 1: request public key PK of entity 2
Step 4: public key PK of entity 2
Entity 1
Entity 2
Step 5: send message M encrypted with public key PK of entity 2

Figure 1.1: Public Key Infrastructure

given to perform. Moreover, the entities like system administrators form a hierarchy to ease the maintenance of grid. Hence, hierarchical access control may also be provided as security functionality in addition to the above list of functionalities [6].

As group-oriented communication have been increasingly facilitating many emerging applications like video conferencing that require packet delivery from one or more sender(s) to multiple receivers secure group communication (SGC) has attracted much attention. Grid Security Infrastructure (GSI) [7] is most widely used for providing security in grid; however it has no support for secure group communication. GSS-API has no provisions for group security contexts. If n grid entities want to communicate among themselves using the usual mechanism of point to point public key cryptographic algorithms, there would be a requirement of n*(n-1)/2 key pairs. Each member would have to keep track of n keys. If the same message has to be sent to all the remaining nodes n-1 encryptions and n-1 messages have to distributed. Suppose, if all the n members have a shared secret key which only the group members know, if a member wants to send a message to all other entities only 1 encryption and n-1 messages have to distributed. Therefore, a lot of computation overhead is removed. Since, a grid is established for performance improvement, if much overhead, as is involved in the point to point security schemes, the performance would degrade very much [8]. There are many methods which are based on group signatures, message digests and public key encryption that address the issue of secure group communication (SGC) [9]. However, the most widely used scheme is to use a group key which is known to all the group members and to use a symmetric encryption scheme. AES is one of the widely used symmetric encryption algorithms [10].

1.2. Motivation

This section describes what is group rekeying, why it is important and why an efficient rekeying scheme is desired in grid.

Providing key secrecies is an important task that a secure group communication mechanism must handle. There are three aspects of key secrecies: forward secrecy, backward secrecy and perfect forward secrecy. The forward secrecy ensures that a new joining member cannot use the new key to decrypt all messages which have been encrypted with the previous key(s). The backward secrecy ensures that a leaving member cannot use the previous key(s) to decrypt all messages encrypted with the new key. The perfect forward secrecy ensures that a compromise of a long-term key seed that generates the present short-term key(s) cannot deprive the secrecy of other previous short-term keys which have been generated by the compromised long-term key. When a member joins/leaves, the group key has to be changed so as to make sure that the key secrecies are satisfied. This process is called rekeying.

A group key management scheme is responsible for key secrecies, key independence and rekeying. Providing a secure group communication mechanism that is scalable does not simply suffice for grid environment. The grid user base is large and dynamic [2]. Hence, the group membership changes are often. So, the efficiency of rekeying during a member change is a key parameter in deciding the performance of the secure group communication scheme. Moreover, the uncertainty in user behavior also makes rekeying difficult [11]. Hence, any SGC mechanism in grid has to be scalable as well as should provide efficient rekeying strategy.

1.3. Problem Statement

The aim is to design and implement a secure, scalable and efficient mechanism that facilitates group communication in grid environment.

A few security loop holes have been identified in existing mechanisms and mechanisms have been developed to remove them.

Some of the encoding schemes consume high bandwidth while needing less computation costs, some of the encoding schemes consume less bandwidth while computation costs are high. Hence, a combined threshold based encoding scheme has been developed so that the encoding scheme can be configured as per the bandwidth and computation limits.

1.4. Organization of Dissertation

This thesis report comprises of six chapters including chapter 1 that introduces the secure group communication, grid computing, motivation, and states the problem. The rest of the dissertation report is organized as follows:

- Chapter 2 provides background and literature reviews on various existing mechanisms that aid secure group communication in grid.
- Chapter 3 provides a detailed description of the various proposed mechanisms and a few methods to compromise the security of some of the mechanisms have been discussed.
- Chapter 4 gives the brief description of the implementation of the proposed scheme.
- Chapter 5 details various simulation experiments conducted and interprets the various results obtained.
- Chapter 6 concludes the work and gives the directions for future work.

CHAPTER 2
BACKGROUND AND LITERATURE REVIEWS

Various mechanisms that have been proposed to facilitate the secure group communication in grid environment have been discussed and classified in this section.

Some of the existing mechanisms are:

An authenticated Diffie-Hellman key-exchange protocol (DDHP) which considers the dynamic situation when the members join/leave the group and the situation that processes have sub-processes is proposed by Chen Lin, et al. [3].

A grid security architecture based on tunable group key agreement (TGKA) is presented by Rajesh Ingle, et al. [2]. TGKA considers leave time information and application class information.

A batch rekeying scheme (BEBS) based on exclusion basis system (EBS) has been proposed by Chi-Chun Lo, et al. [10]. The exclusion basis system (EBS) provides a framework for supporting group key management, especially in a large size network.

Zou Xukai, et al. [7] used a novel concept/construction of Access Control Polynomial (ACP) and one-way functions. The ACP is used to encode the group key so that only those members who are in the group currently can extract it efficiently and those who are not in the group cannot obtain the correct key. The scheme can be used for securing different Grid based services. DLKM consists of two levels:

- The first to solve the problem of Dynamic Groups and Key Distribution efficiently
- The second designed for HAC, which is built upon the first level

A rekeying operation when a member joins or leaves is simply one polynomial multiplication/division. So, the rekeying is independent of the group size. If it were proportional to group size, then the method would not be scalable. However, since it uses polynomial multiplication/division, it is computationally intensive.

Yunfa Li, et al. [13] has proposed a scalable service scheme (SSS) for secure group communication in grid. In the service scheme, a series of methods and strategies are presented, such as the initialization methods for group member, administrative domain and virtual organization, the key distribution strategy and the rekeying strategy. In order to improve the scalability of this service scheme, the services for a group are logically divided into two hierarchical levels, which is in accordance with the characteristics of group communication in grid. In addition, in order to show the efficiency and the scalability of the service scheme, simulation experiments are done. The results show that the service scheme is efficient and scalable. Thus, the service scheme can satisfy the requirement of people in large-scale, dynamic grid environment.

Exploiting the orthogonal principle A.John Prakash, et al. [14] propose a provably secure n-way cryptosystem (Multicrypt).

The P-LEASEL [15] algorithm is scalable because of its hierarchical nature and it provides a very good authentication since it uses Kerberos protocol for authentication. More over here, we can use any symmetric algorithm for encrypting and decrypting without going into difficulties of generating a complex group key. The major advantage of P-Leasel model is that it divides the group into many subgroups and divides the overhead of handling the management of the group. In P-Leasel it is assumed that users are chosen according to trust. The actual evaluation of trust combined with P-LEASEL is proposed in [16].

EAB [17] is also a scheme that encodes group key like DLKM [7]. In DLKM the encoding is done using one way functions and polynomial arithmetic. EAB is based on integer arithmetic. So, it is simple to implement and less computationally intensive. Moreover, during a member join, the group key need not be changed.

M Venkatesulu [18] has proposed PMM, a method which is also based on integer arithmetic. It is based on the same assumptions of RSA algorithm that given a number, finding out whether it is prime, is computationally easy and given a number, finding out the prime factors of that number is computationally difficult. So, the proposed encoding scheme secures the group key. During a member join, the group key need not be recalculated and more over, the rekeying overhead on member join/leave is independent of the group size just as the DLKM method and this method uses integer arithmetic instead of complex polynomial arithmetic used by DLKM and hence it is efficient.

2.1. Classification of Existing Mechanisms for SGC in Grid

The classification of various existing mechanisms for secure group communication in grid can be done based on the organization of group, kind of group key management and on the kind of group key encoding strategy.

Based on whether entire group key management is done by single KDC, a group of KDC or by all the members of the group, the mechanism can be divided into centralized, decentralized and distributed respectively. The entities in grid may have very different functionalities system administrators, sensors, resources etc. Most of the entities are such that they have dedicated roles to perform in the grid and communicate with only those entities that are required in the task that it is given to perform. So, usually decentralized schemes based on the division of the group with respect to the roles tend to decrease the number of rekeying scenarios per sub group. The classification schema is shown in figure 2.1.

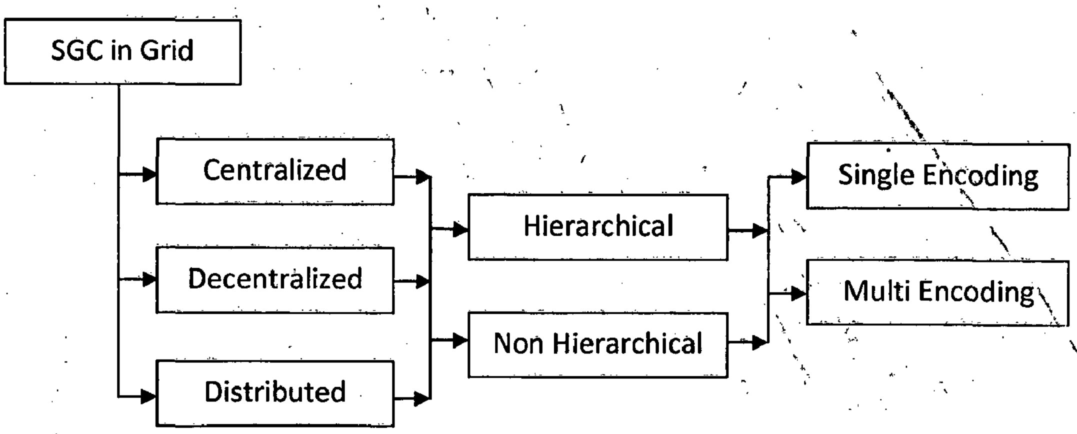


Figure 2.1: Classification scheme for SGC mechanism in grid

Table 2.1: Classification of existing SGC mechanism in grid

Mechanism	Classification
DDHP [3], TGKA [2], BEBS[10]	Centralized, Non Hierarchical, multi encoding
DLKM[7], EAB[17]	Centralized, Hierarchical, single encoding
SSS [13]	Distributed, Hierarchical, multi encoding
Multicrypt [14], PMM[18]	Centralized, Non Hierarchical, single encoding
P-Leasel [15]	Decentralized, Hierarchical, multi encoding

Moreover, the entities like system administrators form a hierarchy to ease the maintenance of grid. Based on whether the group is considered as a hierarchy or all the entities considered equal, the existing mechanisms may be classified as hierarchical or non hierarchical respectively. Hierarchical schemes tend to decrease the number of rekeying scenarios. They pose little overhead per entity that is responsible for maintenance of the group. P-Leasel [15] model utilizes this property of grid and divides the group into sub groups, hence, decreasing the number of rekeying scenarios.

Based on whether the result of encoding the group key is same for all the group members or different, the mechanism can be further divided into single encoding, multi encoding respectively. It can be seen that the single encoding schemes have advantage in the number of bytes present in the result of the encoding since it need not have to consider other group members but computation wise it is inefficient as for each member a separate computation has to be made. The classification of various mechanisms discussed above according to the classification scheme discussed is shown in table 2.1. Some of the mechanisms have been modified to make them computationally efficient and secure. A brief overview of the mechanisms is needed to understand the proposed work.

2.2. P-Leasel Model

In this section, the various entities of the P-Leasel model have been discussed and a brief overview of the various algorithms in the model has been discussed [15]. The advantages and disadvantages of the model have been discussed.

2.2.1. Advantage of P-Leasel

Due to the division of the group into subgroup, the number of rekeying scenarios to be handled by a single KDC has been decreased. Since, the grid population is large, the arrival of join/leave events can be considered poison. According to traffic theory, the division of the requests uniformly will cause the arrival rate to decrease as shown in figure 2.2.

2.2.2. System components

The various entities involved in the model and their roles are:

Controller: It is responsible for initial authentication of a user node. There is a GACL (Group access control list), the private keys and other pertinent details, needed for authentication.

Deputy Service Provider: There is a SACL (Sub group Access Control List) at each DSP for determining if a user is eligible for a service. After initial authentication by the Controller; the member is allocated to a subgroup under a Deputy Controller. The deputy controller decides the rank of all the members in the subgroup. The first ranked member is designated

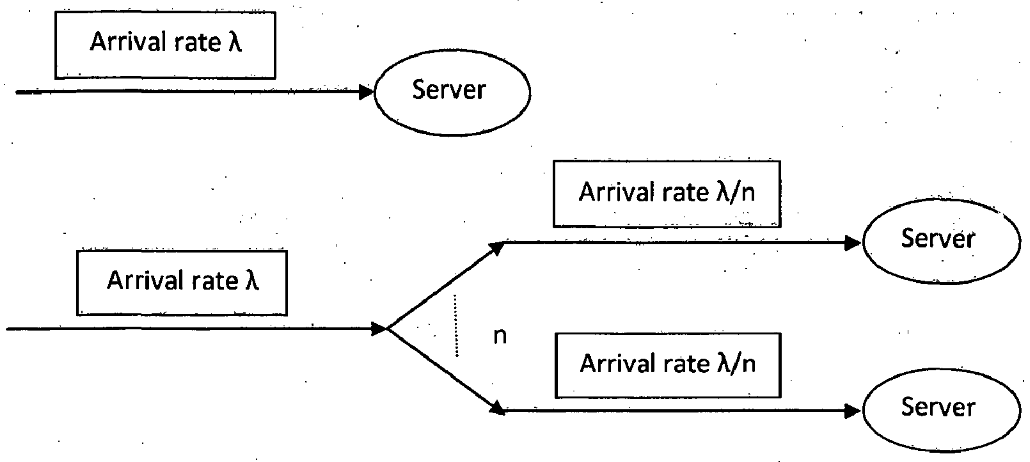


Figure 2.2: Rekeying rate decreases as the number of subgroups increases

as the Leader and is entrusted with the responsibility of key generation and distribution. The deputy controller alone knows the identity of the leader and sender anonymity is achieved by hiding the identity of the leader from the other members of the sub group.

Leader: The leader is responsible for key management and distribution, securing data transmissions using the group key. The identity of the leader is kept secret, known only to the DSP which selects it. Even the node vested with the Leader responsibility is not eligible to know that Leader role is currently performed by itself. The leader is dynamically selected.

The other entities involved in the model are user nodes which are the service requesters and service nodes, which are the service providers. Figure 2.3 shows how a member join is handled.

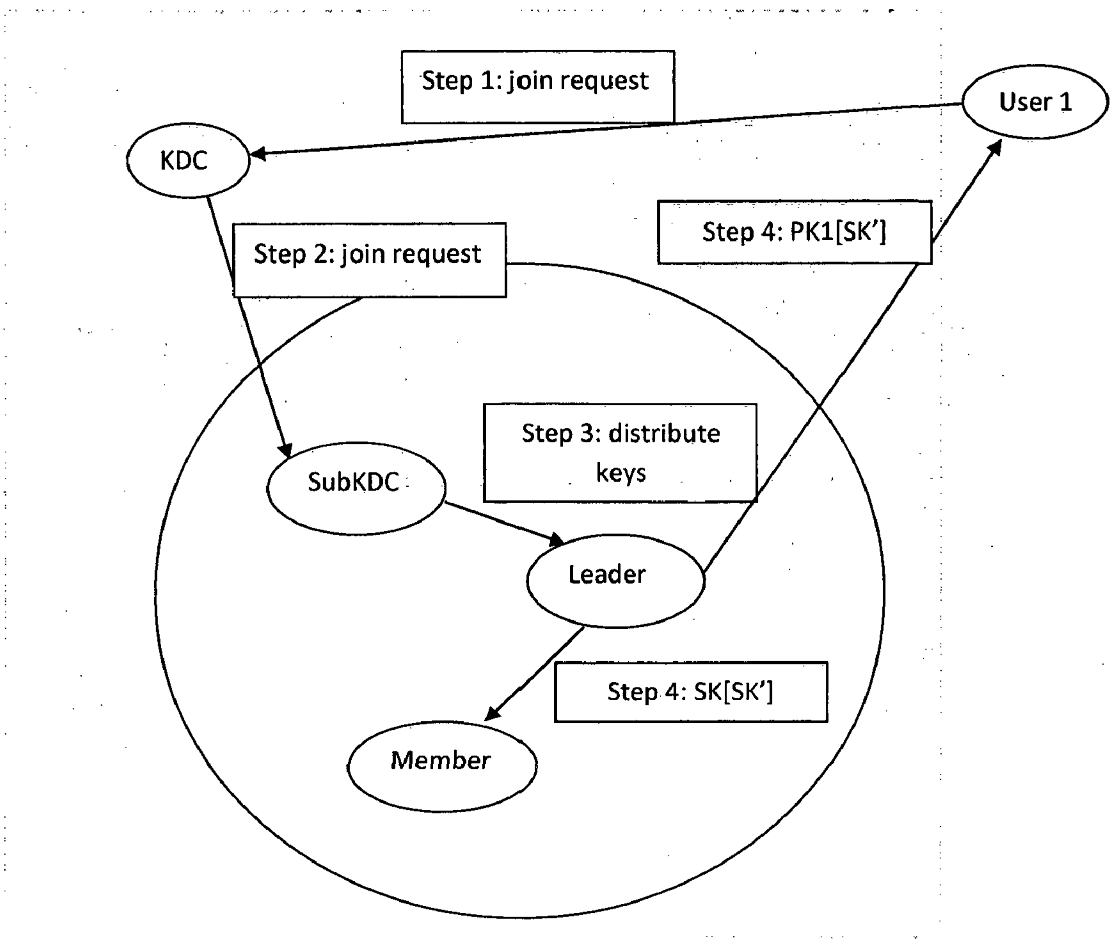


Figure 2.3: Member join handling using P-Leasel algorithm

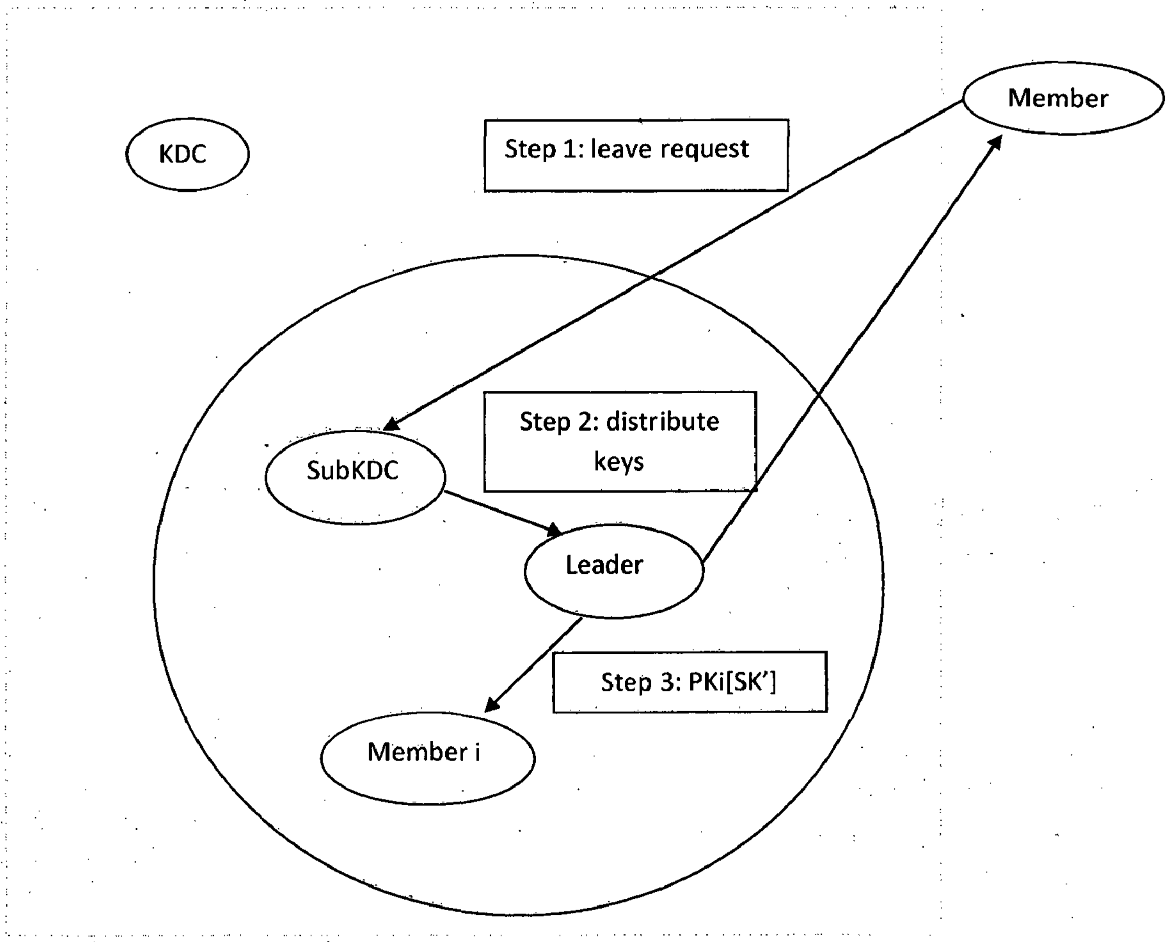


Figure 2.4: Rekeying during member leave in P-Leasel algorithm

From the figure 2.4, it can be seen that during a member leave the number of encryptions to be done is very high. Hence, a huge computational load is placed on the leader node. This is further verified from the results in chapter 5.

2.3. Euclidian Algorithm Based Key Computation Protocol (EAB)

EAB [17] is also a scheme that encodes group key like DLKM [7]. In DLKM the encoding is done using one way functions and polynomial arithmetic. EAB is based on integer arithmetic. So, it is simple to implement and less computationally intensive. Moreover, during a member join, the group key need not be changed. The scheme is very simple, each user M_i is assigned a permanent personal security number PSN_i and the group key is encoded using equation (1) and (2) in table, where as it can be recalculated by using (3) in table 2.3

2.2.1. Assumptions

It is assumed that certificates are present and when a permanent personal security number (PSN) is issued to a member, it will be embedded into the certificates issued at the time of joining. It is assumed that the member mi would recognize the pair (a_i, b_i) meant for it.

2.2.2. Algorithm

Consider that the group consists of 'n' members (M_1, M_2, M_3...M_n) and the current group key is K. User Node with Unique ID = UID_i is depicted as M_i. The PSN of M_i is termed as PSN_i. For a group of users participating in a grid service, the Key Distribution Manager (KDM) will generate (a_i, b_i) pairs for each M_i using PSN_i and equations (1) and (2) from table 2.2.On receiving the pair (a_i, b_i) , M_i calculates group key K using equation (3) from table 2.2. Figure 2.5 and figure 2.6 show rekeying during member join/leave in EAB.

Table 2.2: Equation table for EAB

1. ai = K / PSNi
2. bi = K mod PSNi
3. K = ai*PSNi+bi

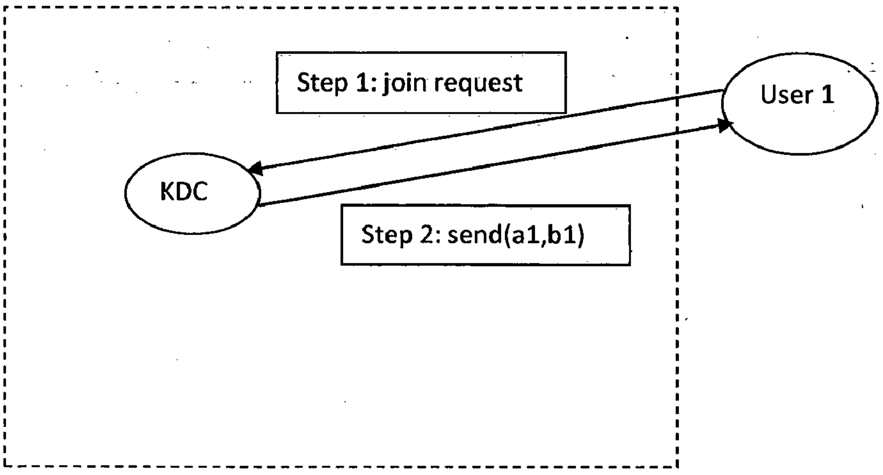


Figure 2.5: Rekeying during member join in EAB

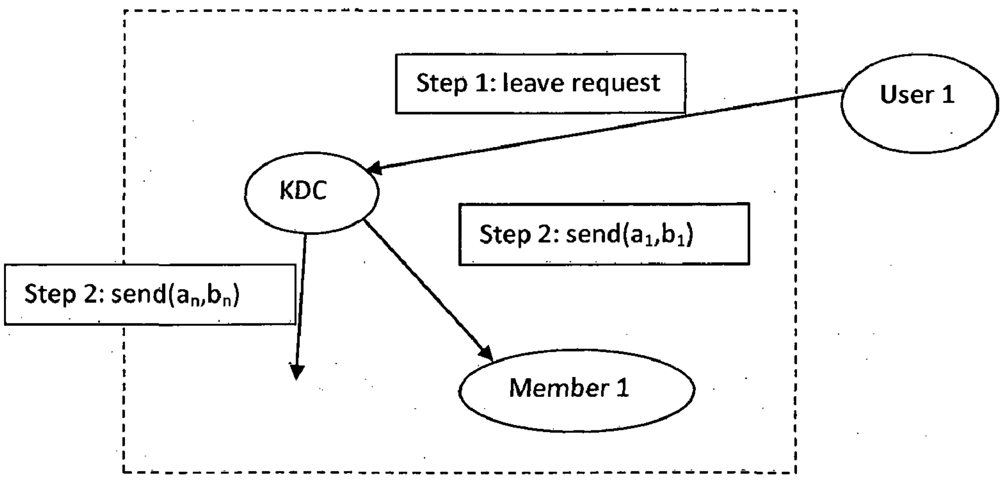


Figure 2.6: Rekeying during member leave in EAB

2.4. Prime Modulus Method (PMM)

M Venkatesulu [18] has proposed a method which is also based on integer arithmetic. It is based on the same assumptions of RSA algorithm that given a number, finding out whether it is prime, is computationally easy and given a number, finding out the prime factors of that number is computationally difficult. So, the proposed encoding scheme secures the group key. During a member join, the group key need not be recalculated and more over, the rekeying overhead on member join/leave is independent of the group size just as the DLKM method and this method uses integer arithmetic instead of complex polynomial arithmetic used by DLKM and hence it is better.

2.4.1. algorithm in PMM

Consider there are n members in the grid system N= (m1, m2......mn) and the group key is K. Each Member mi will be assigned a Pi, a large prime number > K. The following notations in table 2.3 are used to simplify the understanding

Table: 2.3: Notation table for PMM

1. P(K) = P1*P2*P3*...*Pn
2. P(K,i) = P(K)*Pi
3. P'(K,i) = P(K) / Pi
4. G(K) = P(K) + K
5. G(K,i) = P(K,i) + K
6. G'(K,i) = P'(K,i) + K
7. K = M mod Pi

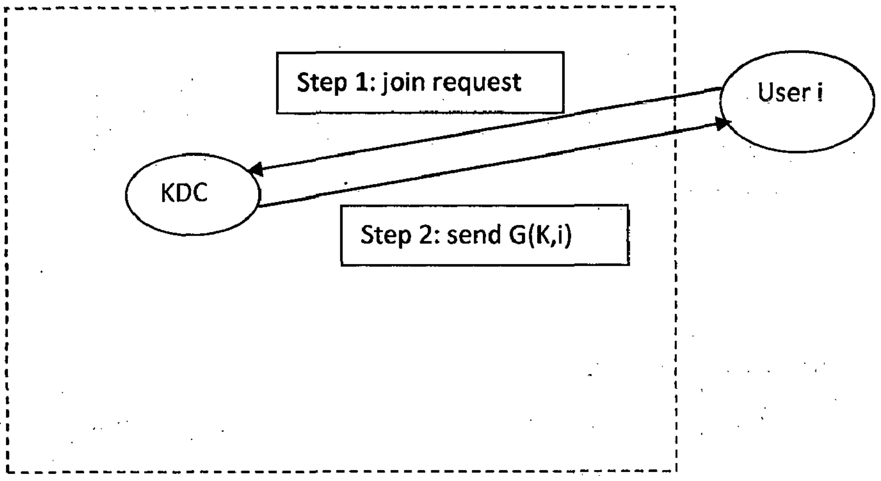


Figure 2.7: Rekeying during member join in PMM

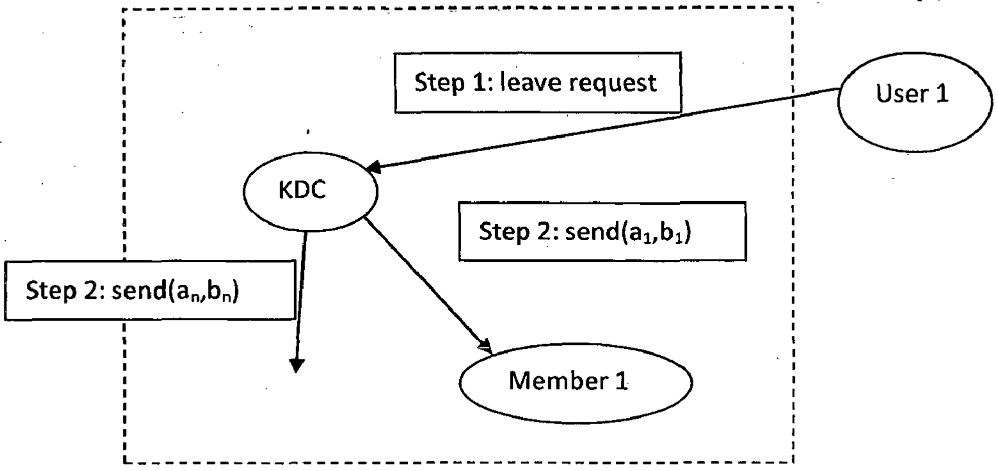


Figure 2.8: Rekeying during member leave in PMM

Advantages:

1. During a member join no need to generate new group key
2. During a member leave only one division and addition operation is required. Computation cost is independent of group size.

2.5. Research Gaps

2.5.1. P-Leasel Model:

Although the model utilises the architecture of grid to its advantage, it uses traditional encryption scheme to distribute group keys to group members. Although during a member join only two encryptions are to be done at KDC but when a member leaves the system , the new group key has to be sent to each user encrypting it with the user's private key, to ensure backward key secrecy.

2.5.2. EAB:

1. In case of a member leave, n-1 number of (ai,bi) pairs have to computed followed by n-1 message send.
2. The encoding strategy is not secure enough to keep PSN secret. Since PSN is a permanent number assigned to a user, all the group keys when one of the compromised user nodes is in the group can be found out. Moreover, if any user node whose PSN is not known to the malicious node has joined the group, its PSN can be found out. This leads to more compromised user nodes. So, backward confidentiality is not satisfied.

2.5.3. Prime number method:

1. The size of public message increases as the group size increases. Hence, transmission costs increase.
2. The method is not backward confidential. Until another member leaves from the group after the malicious user of type 1, the malicious user knows the group key. So, the grid system is compromised till another member leaves.

CHAPTER 3
PROPOSED **SOLUTION**

In this section how various existing schemes can be compromised have been discussed. Moreover, the various proposed solutions to make these schemes more secure have been discussed in details.

3.1. Cracking EAB

This section describes how the existing EAB [17] mechanism can be cracked. A malicious user is added to the system. The malicious user node M_i contains the list of compromised user's PSN in compromised user list (CL). The behaviour of the malicious node during various membership events register, join and leave is described in the table 3.2. The notation (k) means equation k from the table 3.1.

Table 3.1: Equations used in cracking EAB

1. $K = a_i*PSN_i+b_i$
2. $PSN_i = (K - a_i) / b_i$

Table 3.2: Behaviour of Malicious node during membership events in EAB

Register	1. Malicious User Node M_i sends register request to KDC. 2. On receiving the PSN_i sent by the KDC to M_i, M_i stores PSN_i.
Join	1. Malicious User Node M_i sends Join request to KDC. 2. On receiving the pair (a_i,b_i), M_i calculates group key K using (1).
Leave	1. Some user node, M_k sends Leave message to KDC. KDC will update K and distribute it. 2. On reception of the pair (aj,bj), if PSNj is in the list CL, go to step 3 else go to step 4. 3. Mi, will calculate the group key K using (1). 4. Mi calculates PSNj using (2) and adds it to list CL.

Once the identity of PSN_j is known to M_i, all the future sessions in which M_j is in the group are compromised. That means, the group key K can be known to M_i when M_j is in the group, although M_i is not a member of the group. Hence, the EAB system is not backward confidential.

3.2. EABv2

An initial attempt on rectifying the security loop hole in EAB has resulted in EABv2.

3.2.1 Design Challenges

The loop hole in the EAB method as described in the previous chapter is in sending the pair (a_i,b_i). Since both a_i and b_j are known to malicious user, it was possible to get PSN_j and hence, compromise the security. The method that is proposed further encodes a_i and b_i into a single variable. Only the KDC and the respective user node will know which variable to use to encode/decode a_i and b_i. The security of the new proposed method lies in that given a prime number, it is possible to find out the next probable prime using miller-rabin test and lucas lehmer test but given the difference between a prime number and its next prime number, it is computationally infeasible to find out either of the prime numbers.

3.2.2 Assumptions

It is assumed that certificates are present and when a permanent personal security number (PSN) is issued to a member, it will be embedded into the certificates issued at the time of joining. It is assumed that the member mi would recognize Ki meant for it. It is also assumed that KDC and all user nodes use the same primality testing algorithm.

3.2.3 Algorithm

Before describing the algorithm, a notation that is commonly used has to be discussed. PSN_{i_Nxt} is the notation used to denote a probably prime number which is the smallest number greater that PSN_i that passes the primality test that is known to both KDC and user node M_i. The following table 3.3 gives the details of the various computations that are used in the scheme. The table 3.4 describes the behaviour of KDC during various membership events and table 3.5 describes the behaviours of user node during various membership events.

Table 3.3: Equations used in EABv2

1. $a_i = K / PSN_i$
2. $b_i = K \bmod PSN_i$
3. $K = a_i*PSN_i + b_i$
4. $a_i = K_i / PSN_i\_Nxt$
5. $b_i = K_i \bmod PSN_i\_Nxt$
6. $K_i = a_i*PSN_i\_Nxt + b_i$

Table 3.4: Behaviour of KDC during various membership events in EABv2

Register	1. On receiving the register request from M_i, a unique prime number PSN_i and sends it to M_i
Join	1. On receiving the join request from M_i, KDC adds M_i to member list and calculates K_i using (1) , (2) and (3) and sends it to M_i
Leave	1. On receiving Leave request from M_i, the KDC will generate a new group key newK 2. M_i is removed from the group member list 3. For every member M_j in the group member list of KDC, it will calculate K_i using (1) , (2) and (3) and sends it to M_j

Table 3.5: Behaviour of User node during various membership events in EABv2

Register	1. User Node M_i sends a register request 2. M_i stores the PSN_i
Join	1. M_i sends Join request to KDC 2. On reception of K_i, M_i will calculate the group key K using (4), (5) and (6)
Leave	1. M_i sends Leave message to KDC 2. On reception of K_i, M_i will calculate the group key K using (4), (5) and (6)

3.2.4 Security Analysis

This subsection describes the how secure the EABv2 is. Various notations used in the security analysis of EABv2 have been shown in table 3.6. Assume malicious node is M_i and member node is M_j, KDM sends $K_{(i,k)}$ and $K_{(j,k)}$. Mi captures both. It calculates K_k using equations (4), (5) and (6) in table. M_i calculates $P_{(j,k)}$ using equation (1) of table 3.7.

Now M_i knows $P_{(j,k)}$. Using the same process, M_i can repeatedly compute $P_{(j,k')}$, $P_{(j,k'')}$ etc and calculate GCD of all $P_{(j,k)}$ s to get D_j. Once the difference D_j is known we can find out $a_{(j,k)}$ using equation (2). Since,

$b_j = K \bmod PSN_j$

- ⇨ $0 <= b_j < PSN_j$,
- ⇨ $a_j*PSN_j <= a_j*PSN_j*b_j < (a_j+1)*PSN_j$
- ⇨ $a_j*PSN_j <= K < (a_j+1)*PSN_j$
- ⇨ $K/(a_j+1) < PSN_j <= K/a_j$

From (14) we can get the range in which PSN_j is present. As more and more a(j,k) are revealed the range will be narrowed down and in that small range, since it is known that PSN_j is prime, calculate PSN_{j_Nxt} -PSN_j and if the number matches with D_j means that the PSN_j is

Table 3.6: Notations used in security analysis of EABv2

$K_{(k)}$	Group key generated for i th member leave
$K_{(j,k)}$	$a_{(j,k)}*PSN_{j\_Nxt}+b_{(j,k)}$
D_j	$PSN_{j\_Nxt} - PSN_j$
$P_{(j,k)}$	$a_{(j,k)}*D_j$

Table 3.7: Equations used in security analysis of EABv2

Equations
(1)$P_{(j,k)} = K_{(j,k)} - K_k$
(2)$a_{(j,k)} = P_{(j,k)} / D_j$

Table 3.8: Behaviour of malicious node during various membership events in EABv2

Register	1. Malicious User Node M_i sends register request to KDC 2. On receiving the PSN_i sent by the KDC to M_i, M_i keeps PSN_i for future reference
Join	1. Malicious User Node M_i sends Join request to KDC 2. On receiving the pair K_i, M_i calculates group key K using (1)
Leave	1. M_l sends Leave message to KDC *Let* k *be the user node that M_i want to attack* 2. On reception of the pair K_j, if j = i, go to step 3 else if j = k, go to step 4, else stop 3. M_i, will calculate the group key K using (1) 4. Calculates Dj and the range of PSNj. Finds out the PSNj by the method described below

compromised. So, we can crack PSN_j. So, the proposed method is not as secure as to keep the permanent private secret number a secret permanently. Table 3.8 shows the behaviour of malicious node that cracks EABv2.

3.3. EABv3

EABv2 is perfectly hiding the identity of b_i, but the problem is that given sufficient information the secrecy of a_j is not maintained.

3.3.1 Design challenges

So, one of the approaches that was thought to be more secure was to increment the term PSN_{i_Nxt} with every section so that taking repeated GCD would not help in knowing D_j and hence a_j. However, both KDC and user node have to keep track of how many leave events the user has encountered till now. This number should match for both user node and KDC. Any error may lead to failure of the protocol. So, an alternative method is proposed which uses the fact that only KDC and M_i know the value of P_i. So, an alternative method is proposed which makes use of the fact that only KDC and M_i know the value of PSN_i.

3.3.2 Assumptions

It is assumed that certificates are present and when a permanent personal security number (PSN) is issued to a member, it will be embedded into the certificates issued at the time of joining. It is assumed that the member mi would recognize the pair (A_i,B_i) meant for it. It is also assumed that KDC and all user nodes use the primality test.

3.3.3 Algorithm

The table 3.9 gives the details of the various equations that are used in the scheme. The table 3.10 describes a few derivations using the above equations. The notation Ek means Equation k. Let A = (A1,A2,A3..) be a set of arguments and E = (E1,E2,E3...) be set of equations and O = {O1,O2,O3,...} be a set of outputs. The notation A-E->B should be interpreted as a subset of A is the RHS of E1 and it has O1 as LHS. Using these notations the derivations in the following table can be interpreted. Using the above derivation table the following table explains how various membership events are handled by the entities. Note that the notation Dk means Detivation number k in table 3.10. Table 3.11 describes various membership events in KDC in EABv3. Table 3.12 describes various membership events in KDC in EABv3.

Table 3.9: Equations used in EABv3

(1) ai = K / PSNi	(2) bi = K mod PSNi
(3) Ai = ai + PSNi	(4) Bi = bi + PSNi
(5) ai = Ai – PSNi	(6)bi = Bi – PSNi
(7) K = ai*PSNi + bi	

Table 3.10: Derivations used in EABv3

Derivation Number	Derivation
(1)	(K,PSNi)-(E1,E2)->(ai,bi)-(E3,E4)->(Ai,Bi)
(2)	(Ai,Bi,PSNi) –(5,6)-> (ai,bi,PSNi) -(7)-> K

Table 3.11: Behaviour of KDC during various membership events in EABv3

Register	1. On receiving the register request from M_i, a unique prime number PSN_i and sends it to M_i
Join	1. On receiving the join request from mi, KDC adds M_i to member list and calculates the pair (A_i,B_i) using D1 and sends it to M_i
Leave	1. On receiving Leave request from mi, the KDC will generate a new group key newK 2. M_i is removed from the group member list 3. For every member M_j in the group member list of KDC, it calculates the pair (Ai,Bi) using D1 and sends it to M_i

Table 3.12: Behaviour of User node during various membership events in EABv3

Register	1. User Node M_i sends a register request 2. User node M_i stores the PSN_i
Join	1. User Node M_i sends Join request to KDC 2. On reception of K_i, M_i will calculate the group key K using D2
Leave	1. M_i sends Leave message to KDC 2 On reception of Ki, mi will calculate the group key K using (6), (7) and (8)

3.3.4 Security Analysis

Given K and PSN_i it is easy to compute A_i, B_i using D1. But given A_i and B_i it is difficult to compute either K or PSN_i for a member M_j who may be an internal attacker or external one. The notation K_i should be interpreted as K_i is the group key for some session i.

$$K_1 = a_{(i,1)}*PSNi+b_{(i,1)} \quad (1)$$

$$A_{(i,1)} = a_{(i,1)} + PSN_i \quad (2)$$

$$B_{(i,1)} = b_{(i,1)} + PSNi \quad (3)$$

$$K_2 = a_{(i,2)}*PSN_i+b_{(i,2)} \quad (4)$$

$$A_{(i,2)} = a_{(i,2)} + PSN_i \quad (5)$$

$B_{(i,2)} = b_{(i,2)} + PSN_i$ (6)

A malicious user M_j will use the above equations and at most by subtraction it can find the difference between a_i and corresponding b_i using (2) and (3) or (5) and (6)but cannot find a_i and b_i individually. Moreover, since we have added PSN_i, not multiplied, taking GCD would not help. So, the identity of PSN_i will also remain secret. At most M_j would know the differences between a_i, b_i etc but never would it get to know any a_i or b_i or PSN_i

3.4. Cracking PMM

The table 3.13 contains some useful equations and the derivation table explains some of the derivations. Table 3.14 describes the behaviour of the malicious user node during various membership events.

Table 3.13: Derivations used in PMM

Derivation Number	Derivation
1	1. (M,P_i)-E1->K 2. (M,K) –E2->P(K) 3. $(P(K),P_i)$-E3->P'(K,i)

Table 3.14: Malicious user behaviour during various membership event in PMM

Register	1. On receiving the register request from M_i, KDC will generate a unique prime number P_i(>K) and sends it to M_i.
Join	1. User Node M_i sends Join request to KDC 2. M_i stores P_i on reception 3. On reception of M, mi will calculate the group key K using E1 and P'(K,i) using D1
Leave	1. mj sends Leave message to KDC 2. On reception of M, mj, will calculate the group keyK using E4

3.5. Dual Prime Modulo Method (PMMv2)

The malicious user calculates the group key K because he knows the product P(K) and exactly what the product is going to be when the user leaves the system because P_1 would be divided from P(K) and the new encoded key would contain P(K) is P'(K,i). This happens because the term P(K) is solely based on the user's P_i. Moreover, the user U_i knows exactly what will be divided from P(K) to get P'(K,i). So, the PMMv2 uses 2 primes per user, one is assigned to the user permanently at the registration time and the other is kept secret but included in the calculation of the product P(K). The following notations in table 3.15 will be used to describe the method. How various membership events are handled by KDC is described in table 3.16 and how various membership events are handled by KDC is described in table 3.17.

Table 3.15: Notations used in PMMv2

Notation	Mathemetical equivalent
P(K)	P11*P12*P21*P22*...*Pn1*Pn2
P(K,i)	P(K)*Pi1*Pi2
P'(K,i)	P(K) / (Pi1*Pi2)
G(K)	P(K) + K
G(K,i)	P(K,i) + K
G'(K,i)	P'(K,i) + K

*Where *,/,+ stand for arithmetic multiply divide and add respectively*

Table 3.16: Behaviour of KDC during various membership events in PMMv2

Register	1. On receiving the register request from mi, KDC will generate a unique prime number Pi(>K) and sends it to mi.
Join	1. On receiving the join request from mi, KDC will add mi to its member list. 2. KDC generates the new public message M as M = G(K,i) and sends it to mi
Leave	1. On receiving Leave request from mi, the KDC will generate a new group key newK 2. mi is removed from the member list 3. KDC calculates the public message M as M = G'(newK,i) and sends it to all group members

Table 3.17: Behaviour of User node during various membership events in PMMv2

Register	1. User Node mi sends register request to KDC 2. Mi keeps the received Pi for future reference
Join	1. User Node mi sends Join request to KDC 2. mi stores Pi on reception 3. On reception of M, mi will calculate the group key K using (1)
Leave	1. mi sends Leave message to KDC 2. On reception of the pair M, mj, will calculate the group key K using (1)

3.4.1 Security analysis

Consider a group of n members formed using PMMv2 being attacked by the malicious user as described in the previous section. The malicious node Mn calculates the product P'(K,i) after he left the group as

P' = P11*P12*.....Pn-11*Pn-12*Pn2

Whereas the actual,

P'(K,i) = P11*P12*.....Pn-11*Pn-12

Since, G(K)mod P' = (P'(K,i) + K) mod P' and P'(K,i) mod P' is not zero, we can protect the group key from being compromised

3.4.2 Compromising Security

Consider n members in the group M = $\{M_1, M_2 ... M_n\}$. Consider that M_1 is a malicious user node. Whenever a membership leave takes place, the KDC sends the new public encoded message $G(K_k)$ (K_k is the notation for group key at session k), it will calculate $P_k' = P(K)/P_i$. Now consider the scenario when Mi left G(Kk-1) was sent by KDC and when Mn left G(Kk). Suppose, between the two leaves, a user mn' joins the group. Now, the Pk-1' and Pk' are both available

P_{k-1}' = P11*P12*...*Pn-11*Pn-12 *Pn1*Pn2

P_k' = P11*P12*...*Pn-11*Pn-12 *Pn1'*Pn2'

Figure 3.1: Shows GCD terms in contiguous sessions of PMMv2

As it can be seen in the above figure, the user M_1 now knows the product of primes that is assigned to Mn'. Now, although it cannot factorize and find out Pn1 it can now leave the system and get the correct group key K as K = G(K) mod (Pn1'*Pn2') since it is included in the calculation of P(K). Untill and unless the user Mn' leaves, the malicious user can continue to hold the group key K.

Now, even if we increase the number of primes assigned to the user, it would still not protect the newly joined users.

3.6. Rotation Based Prime modulo Method (RPMM):

The flaw in security for the methods PMM and PMMv2 is that, in the calculation of P(K), both methods use only the credentials assigned to user. As a result, when a user leaves all the credentials assigned to the user have to removed. And when a user joins they have to be included. So, it is easy to know for those members who are already present in the group to know these credentials. Since, the methods defined so far are multiplying these credentials, taking GCD of different number of P(K) and trying to separate the credentials. An ideal scheme aims to provide security by hiding these credentials no matter how many P(K) the user captures. That is, if the user tries to obtain GCD and the different terms, using n number of P(K), the method should be able to hide the credentials for all n = 1,2,3,... to till the user stays in the system.

Rotation Primes Method consists of a set of prime numbers N = {RP1,RP2,...}. Consider, initially there are n members in the group M = {M1,M2,..Mn}. The following table

Table 3.18: Notations used in RPMM

Notations	
Np	Number of primes in N
RP_i	ith prime in set N
RP	$RP_1*RP_2*..RP_{Np}$
RPi’	RP/RP_i
P(K,i)	$P_1*P_2*...*P_n*RP_i$’
P(K,i,j)	$P_1*P_2*...*P_n*P_j*RP_i$’
P’(K,i,j)	$P_1*P_2*...*P_{j-1}*P_{j+1}*...*P_n*RP_i$’
G(K,i)	P(K,i) + K
G(K,i,j)	P(K,i,j) + K

describes various notations used. suppose the current public key message is G(K,i). When a user Uj joins the group the new publc message becomes G(K,i,j).Rotating prime method provides both for a limited number of sessions. But what should we do when a user stays for a long enough time in the system, then we cannot provide varying GCD anymore. Now the system enters a compromised state. So, the method makes the system secure for a limited number of member leaves. However, the prime number method has an advantage that can still be utilised. Instead of assigning permanent prime numbers, assign for a long duration. Whenever the system is compromised, the primes would be refreshed using private keys and encryption.

Table 3.19: Behaviour of KDC during various membership events in RPMM

Register	1. On receiving the register request from mi, KDC will generate a unique prime number Pi(>K) and sends it to mi.
Join	1. On receiving the join request from mi, KDC will add mi to its member list. 2. KDC generates a prime number Pi (> K) and sends it to mi 3. KDC generates the new public message M as M = G(K,i) and sends it to mi
Leave	1. On receiving Leave request from mi, the KDC will generate a new group key newK 2. mi is removed from the member list 3. KDC calculates the public message M as M = G'(newK,i) and sends it to all group members 4. If the number of leave events that mi is more than the number of primes in rotation, all the user mj in member list will be sent newPj encrypted with their private key, a new group key is computed and the public rekeying message M which is calculated with the new primes and new rotating primes as shown in above figure 3.1.

Table 3.20: Behaviour of User node during various membership events in RPMM

Register	1. User Node mi sends register request to KDC 2. Mi keeps the received Pi for future reference
Join	1. User Node mi sends Join request to KDC 2. mi stores Pi on reception 3. On reception of M, mi will calculate the group key K using (1)
Leave	1. mi sends Leave message to KDC 2. On reception of M, mj, will calculate the group key K using (1) 3. If newP is sent, the old Pi is removed and the new one is stored as Pi.

3.7. RPMMv2:

As discussed earlier, the security of those members who reside in the system for a long term are the ones that can know the Pi of a user at the cost of his Pi being compromised. A modification in the scheme makes the number of members, whose Pi has changed. The new scheme only updates those Pi, where mi has stayed in the group to witness number of leave events more that the number of primes in rotation.

Table 3.21: Behaviour of KDC during various membership events in RPMMv2

Register	1. On receiving the register request from mi, KDC will generate a unique prime number Pi(>K) and sends it to mi.
Join	1. On receiving the join request from mi, KDC will add mi to its member list. 2. KDC generates a prime number Pi (> K) and sends it to mi 3. KDC generates the new public message M as M = G(K,i) and sends it to mi
Leave	1. On receiving Leave request from mi, the KDC will generate a new group key newK 2. mi is removed from the member list 3. KDC calculates the public message M as M = G'(newK,i) and sends it to all group members 4. If the number of leave events that mi is more than the number of primes in rotation, only those mj who have stayed long enough in the system will be sent a newPj using their private key, a new group key is computed and the encoded public rekeying message M which is calculated with the new primes and new rotating primes as shown in above figure 3.1.

Table 3.22: Behaviour of User node during various membership events in RPMMv2

Register	1. User Node mi sends register request to KDC 2. Mi keeps the received Pi for future reference
Join	1. User Node mi sends Join request to KDC 2. mi stores Pi on reception 3. On reception of M, mi will calculate the group key K using (1)
Leave	1. mi sends Leave message to KDC 2. On reception of M, mj, will calculate the group key K using (1) 3. If newP is sent, the old Pi is removed and the new one is stored as Pi.

3.8. Zone Based PMM (ZPMM)

Another disadvantage of prime number method is that the message that has to be sent during rekeying increases as the group size increases. EAB has the disadvantage that as the group size increases the number of encodings that have to be done increase as the group size increases.

Table 3.23: Behaviour of KDC during various membership events in ZPMM

Register	1. On receiving the register request from mi, KDC will generate a Prime number PSNi (< Pz,i)
Join	1. On receiving the join request from mi, KDC will add mi to its member list. 2. KDC encodes prime number Pi using and sends the (A,B) pair to mi 3. KDC generates the new public message M as M = G(K,i) and sends it to mi
Leave	1. On receiving Leave request from mi, the KDC will generate a new group key newK 2. mi is removed from the member list 3. KDC calculates the public message M as M = G'(newK,i) and sends it to all group members 4. If the zone from which member mi leave has not been refreshed for number of leave events that is more than the number of primes in rotation, a list of zones that also were not refreshed for a long time will be refreshed. The process of how a zone is refreshed is described below

Table 3.24: Behaviour of User node during various membership events in ZPMM

Register	1. User Node mi sends register request 2. It stores PSNi on reception
Join	1. User Node mi sends Join request to KDC 2. mi computes Pi using ... and stores Pi on reception 3. On reception of M, mi will calculate the group key K using (1)
Leave	1. mi sends Leave message to KDC 2. On reception of M, mj, will calculate the group key K using (1) 3. If the pair (A, B) is sent, the old Pi is removed and the new one is stored as Pi.

3.9. P-leasel combined with ZPMM (PZPMM)

Table 3.25: Behaviour of KDC during various membership events in PZPMM

Register	1. On receiving the Register request from mi, KDC will redirect it to the specified SubKDC. Upon successful registration, it will store the userID and the service for which it is registered in its local database.
Join	1. On receiving the join request, if the user is already registered it will be redirected to the respective SubKDC

Table 3.26: Behaviour of Sub KDC during various membership events in PZPMM

Register	1. On receiving the register request from mi, KDC will generate a Prime number PSNi (< Pz,i)
Join	1. On receiving the join request from mi, KDC will add mi to its member list. 2. KDC encodes prime number Pi using and sends the (A,B) pair to mi 3. KDC generates the new public message M as M = G(K,i) and sends it to mi
Leave	1. On receiving Leave request from mi, the KDC will generate a new group key newK 2. mi is removed from the member list

	3. KDC calculates the public message M as M = G'(newK,i) and sends it to all group members 4. If the zone from which member mi leave has not been refreshed for number of leave events that is more than the number of primes in rotation, a list of zones that also were not refreshed for a long time will be refreshed. The process of how a zone is refreshed is described below

Table 3.27: Behaviour of User node during various membership events in PZPMM

Register	1. User Node mi sends register request 2. It stores PSNi on reception
Join	1. User Node mi sends Join request to KDC 2. mi computes Pi using ... and stores Pi on reception 3. On reception of M, mi will calculate the group key K using (1)
Leave	1. mi sends Leave message to KDC 2. On reception of M, mj, will calculate the group key K using (1) 3. If the pair (A, B) is sent, the old Pi is removed and the new one is stored as Pi.

CHAPTER 4
DESIGN & IMPLEMENTATION

GridSim toolkit[19] is used to simulate the grid environment. Since, GridSim is available in java; Java 1.6.0_17 is used as platform. The java installed is 64 bit. Net Beans IDE has been used. The system consists of 3GB RAM. The details of implementations of individual Group Key Management schemes are provided in this section.

4.1. P-Leasel Model

The GridPLeasel module initiates the controller, user nodes, server nodes and deputy service providers and provides controls to initiate the simulation.

The Controller and DeputyServiceProvider, UserNode and ServiceNode modules carry on the same functionalities as a Key Dis. Then, there is AuthDB which is the module for our database model for authentication in Controller and DeputyServiceProvider. Node is an interface which is implemented by ServiceNode and UserNode to know what is the type of the node (Service or user).

The SubGroupInfoInit holds the information for initialising the subgroups, SubGroupInfoMember is the information that a member should know about its subgroup (subgroup id, SubGroupKey, the DSP of the subgroup and the KGM it is given), SubGroupInfoDSP is the information that a DSP holds for maintaining the Subgroup (pleaders, current leader, etc,). The KeyGenerationModule holds the responsibility of generating sub group keys.

The relations between the various important components of the system are shown in fig. 4.1. The solid lines indicate which module inherits a class. The dashed lines show which module implements interface. The dashed and dotted line indicated composition.

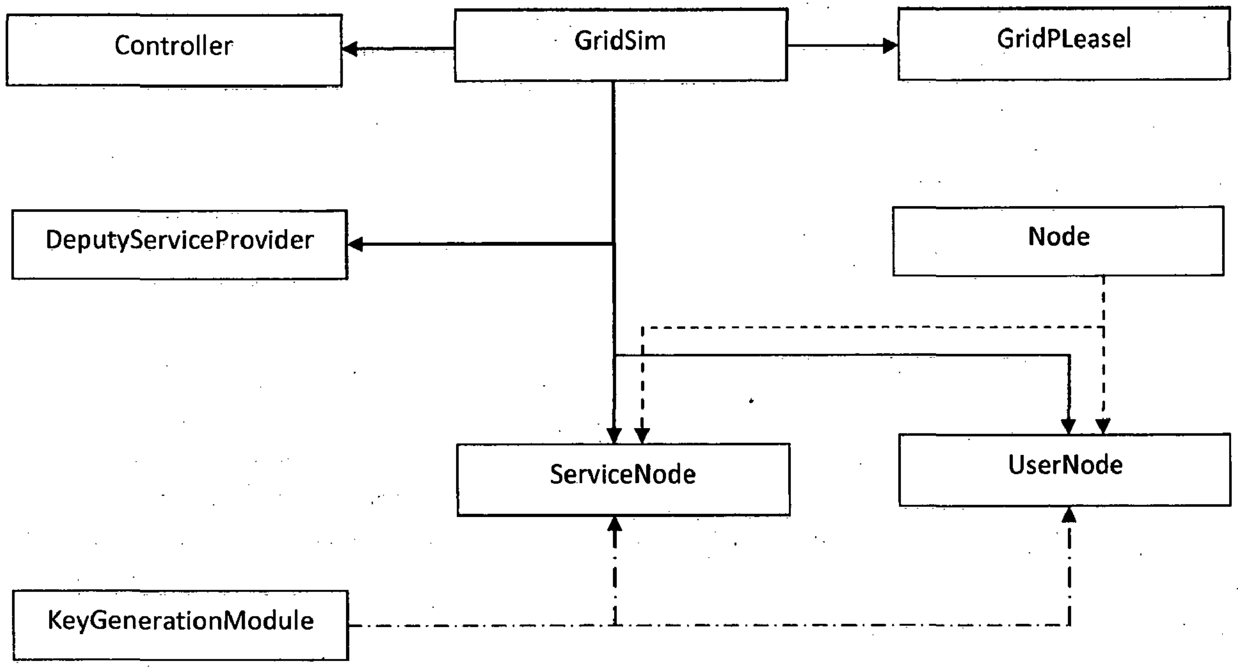


Figure4.1: Components of theP-Leasel system

4.2 EABv1

The various modules and the relation between them is shown in the following figure. The GridSim module is a part of the gridsim toolkit.

The description of various modules is as follows

4.2.1 AES

This module is responsible for encryption and decryption. The key length that is being used is 128 bit.

Table 4.1: AES module

Static variables:
static int KeySize = 16;//The key size in bytes
private static byte[] iv

={0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00} ;//Initialization vector

Instance Variables:
int nEncrypt;// number of encryption
int nDecrypt;//number of decryption
long tEncrypt// time spent in encryption in nano seconds
long tDecrypt;//time spent in decryption in nano seconds
 	
all the instance variables are initialised to 0.

Methods:
 	
byte[] encrypt(byte[] plainText, byte[] pass)
// this method encrypts plaintext using AES method with pass as key. pass should be of 16 byte
 	
byte[] decrypt(byte[] cipherText,byte[] pass)
//this method decrypts cipherText using AES method with pass as key. pass should be of bytes

4.2.2 Key Distribution Centre (KDC)

It is the entity responsible for registration, join and leave of a member in the group. It contains an authentication database which stores <UID, PSN>. It contains the list of group members. It has an AES module that uses 128 bit keys. It has an encoder module that is responsible for encoding the 128 bit group key. It has a Key generation module (KGM) which generates 128 bit group key.

Table 4.2: KDC module

Instance Variables:
private Map<Integer,BigInteger> memPHash;// keeps track of usernodes that are currently in group
private AuthDB authDB;//authentication database

private KGM kgm;//Key generation module

private Encoder encdr;//encoder module

private AES aesM;//AES module

//simulation helper

int nEvents;// no. of events to be kandled by KDC before it shuts down

//statistics

private int nBytesSent;// no of bytes sent by KDC, initialised to 0

StatsKDC stat;// recording statistics

StatsKDC prev;

Methods:

public void body()

// Depending on the kind of event, this method passes control to the function that correc handles the membership event

private void register(Sim_event ev)

//this method handles register membership event

private void join(Sim_event ev)

//this method handles the join membership event

private void leave(Sim_event ev)

//this method handles leave membership event

Void setStat(StatsKDC stat)

//enters the currents statistics information into stat variable

StatsKDC getFinalStat()

//used to get stats from this KDC for the latest membership event

private void send(int id, int event, Object obj)

//since gridsim has support for only point to point communication, this method is implement to facilitate simulations where malicious nodes are involved

4.2.3 Encoder

It is responsible for implementing the encoding schemes.

Table 4.3: Encoder

Static Variables:
int MinBitLen = 32;// minimum bitlength of the prime numbers to be generated
int MaxBitLen = 64;//maximum bitlength of the prime numbers to be generated
Instance Variables:
private Random rnd = new Random();//random seed to generate random length primes
//statistics
private int nEncode = 0;//number of encodings
private long tEncode = 0;//time spend in encodings
Methods:
BigInteger genP()
//generates PSN
byte[][] encode(byte[] SK, BigInteger P)
//encode SK to get pair (ai,bi)
Void setStat(StatsKDC stat)
//enters the currents statistics information into stat variable

4.2.4 Key Generation Module (KGM)

It is responsible for generation of random 128 bit group keys.

Table 4.4: KGM

Static Variables:
Int KeySize = 128;//group key size in bits
Int MinBitLen = 64;//minimum bit length of group key to ensure SK > PSN

Instance Variables:
private Random rnd = new Random();//random seed to generate random length primes
byte[] SK;//group key
//statistics
private int nKeyGen = 0;//number of key generations
private long tKeyGen = 0;//time spend in key genrations

Methods:
 	
BigInteger genSK()
//generates random SK
 	
Byte[] getSK()
//returns SK
 	
Void setStat(StatsKDC stat)
//enters the currents statistics information into stat variable

4.2.5 AuthDB

It is responsible for retrieving information and storing into a XML database of KDC

Table 4.5: AuthDB

Instance Variables:
private String dbname = Simulator.dir+"\\KDC_AuthDB.xml";//file which is used as databa
//statistics
int nInsert = 0;//number of insert operations
long tInsert = 0;//time taken for insert operations
int nRead = 0;//number of read operations
long tRead = 0;//time taken for read operations

Methods:
 	
AuthDB()
//constructor creates database file if already not present
 	
void add(String uID,String PSN)
//inserts the uID and the corresponding PSN in database
 	
String getPSN(String uID)
//retrieves the corresponding PSN for uID as string
 	
Void setStat(StatsKDC stat)
//enters the currents statistics information into stat variable

4.2.6 StatsKDC

This is a module that helps recording all statistics of KDC at one.

Table 4.6: StatsKDC

Instance Variables:
long nBytesSent = 0;
//key generations

```
   int nKeyGen = 0;
   long tKeyGen = 0;
//encryption
   int nEncrypt = 0;
   long tEncrypt = 0;
   int nDecrypt = 0;
   long tDecrypt = 0;
//encoding
   int nEncode = 0;
   long tEncode = 0;
   int nPGen = 0;
   long tPGen = 0;
//database
   int nInsert = 0;
   long tInsert = 0;
   int nRead = 0;
   long tRead = 0;
```

```
Methods:

Void print()
//prints the statistics

Void save()
//saves the statistics in stat file

Void minus(StatsKDC stat)
//subtracts stat from this and resets statistics of stat

Void savePrev(StatsKDC prev)
//copies the statistics into prev
```

4.2.7 MSG

this class is used to facilitate send method in KDC and receive method in usernode.

Table 4.7: MSG

Instance Variables:
String uID;//uID of the user tp which msg is actually meant to be sent
Object obj;//the message

4.2.8 UserNode

It has a Unique ID. It is given a PSN during registration. It consists of a decoder that decodes the group key.

Table 4.8: User node

Static Variables:
Static double wTime;//waiting time
Instance Variables:
protected byte[] PK;// private key
protected AES aesM;//AES module
protected UserDB usrDB;//user database
protected Decoder dcdr; //decoder
protected BigInteger P;// PSN
protected byte[] grpSK;//group key
//extra
protected int mode;//decides user behaviour
//statistics
protected int nBytesSent;
protected StatsUser stat;

protected StatsUser prev;

Methods:
public void body()
// Depending on the user mode, this method uses various methods to make the user node beha
as required
 	
protected void register()
//this method handles registration to group
 	
protected void joinGrp()
//this method handles the join of this node
 	
Private void grpMem()
//when a user remains group node, how it behaves
 	
protected void leaveGrp()
//this method handles leave membership event
 	
Void setStat(StatsUser stat)
//enters the currents statistics information into stat variable
 	
StatsUser getFinalStat()
//used to get stats from this KDC for the latest membership event
 	
private Object receive(int event)
private Sim_event receive()
//these methods handle the message sent by the send method of KDC properly

4.2.9 Decoder

Ddecoder is responsible for retrieving the group key

Table 4.9: Decoder

Instance Variables:
//statistics
private int nDecode = 0;// number of decoding
private long tDecode = 0;//time spent in decoding in nano seconds

Methods:
byte[] decode(byte[] a, byte[] b, BigInteger P)
//retrieves the group key
 	
Void setStat(StatsUser stat)
//updates the statistics in stat

4.2.10 UserDB

This module is responsible for insert or read from XML database of user.

Table 4.10: UserDB

Instance Variables:
private String dbname;//file which is used as database
//statistics
int nInsert = 0;//number of insert operations
long tInsert = 0;//time taken for insert operations
int nRead = 0;//number of read operations
long tRead = 0;//time taken for read operations

Methods: UserDB()

//constructor creates database file if already not present
 	
void add(String uID,String PSN)
//inserts the uID and the corresponding PSN in database
 	
String getPSN(String uID)
//retrieves the corresponding PSN for uID as string
 	
Void setStat(StatsKDC stat)
//enters the currents statistics information into stat variable

4.2.11 StatsUser

StatsUser is similar to StatsKDC and contains same methods with the arguments of type StatsKDC replaced by type StatsUser but the following fields and methods are excluded and a few more are included

Table 4.11: StatsUser

Instance Variables:
//statistics
int nEncode = 0;// number of encoding
long tEncode = 0;//time spent in encoding in nano seconds
int nPGen = 0;// number of prime generations
long tPGen = 0;//time spent in prime number generation

Methods: Void add(StatsUser stat) //adds the statistics to stat

4.2.12 MaliciousUserNode

It also has a Unique ID and PSN. It consists of a decoder that decodes the group key. Additionally, it contains a compromised PSN list that stores the cracked PSN of other users. It also contains a cracker that computes PSN from the pair (ai,bi) and group key K. It is inherited from UserNode.

Table 4.12: Malicious User

Instance Variables:
private HashMap<String,byte[]> aMap;
private HashMap<String,byte[]> bMap;
private Cracker crkr;// cracker that finds out PSN from the information in above maps

Methods:
public void body()
//this method is overridden because grpMem() has to be implemented
 	
private void grpMem()
//this method describes the behaviour of the malicious user node is a group member

4.2.13 Cracker

Cracker finds out the PSN of members

Table4.13: Cracker

Methods: BigInteger decode(byte[] SK,byte[] a,byte[] b) //finds PSN using SK, a and b

4.2.14 Event

A class that contains the events that are used during simulation

Table 4.14: Event

Static Variables:

static final int Register = 1001;
static final int Join = 1002;
static final int Leave = 1003;
static final int A = 1005;
static final int B = 1006;
static final int P = 1007;

4.2.15 Simulator

This module is responsible for initialising the scenario and starting the simulation, and printing the statistics.

Table4.15: Simulator

Static Variables:

static final String dir = "Euclid_v1";
static final String statFile = dir + "\\stat.txt";
static int mode;// which decides simulation mode
static int nGrpMem;// number of users in group

Static methods:
Void simRegistration()
//initialises the entities to be ready for simulation of registration event.
 	
Void simJoin()
//initialises the entities to be ready for simulation of join event.

Void simLeave()
//initialises the entities to be ready for simulation of leave event.
 	
Void simMalicious()
//initialises the entities to be ready for simulation of a malicious user cracking the syste security
 	
Void printStats()
//prints the statistics to console
 	
Void saveStats()
//saves the statistics in file specified by statFile

Methods:
Void setGrpSize()
//how many usernodes should be in the group
 	
Void setMode(int mode)
//sets mode

All the proposed methods EABv2, EABv3, PMMv1, PMMv2, RPMMv1, RPMMv2 and ZPMM have similar modular structure since all these methods are based on Single KDC. PZPMM consists of an additional module SubKDC whose structure is similar to KDC of EABv1 but the functionality achieved is according to that described in previous chapter.

CHAPTER 5
RESULTS

The simulation experiments have been classified into two categories. The first set of experiments demonstrates that the security of some of the existing methods can be compromised and the second set of experiments quantifies the performance evaluation of various implemented mechanisms.

5.1 Security Breach Method Demonstration

5.1.1 Simulation Experiment 1 (EAB)

Simulation Parameters: The user should provide as input, the number of members in the group that must be present in the intial state.

Description: The simulation is carried on as follows. Initially the group contains n-1 members among which one is malicious user node. Now, one of the legitimate user nodes leaves the group. Then KDC generates a new group key using KGM and sends the pairs (ai,bi) to all group members including the malicious node. All the user nodes will calculate group key K on receiving their respective (ai,bi) pair using PSNi. The malicious node will find PSN of all the n-2 user nodes present in the system by following the process described in table earlier and stores them for future reference. Now the malicious user leaves the group and again KDC and the remaining n-2 user nodes repeat the same process as described above but the malicious user knows the group key K. Hence, backward confidentiality is not satisfied. The following figure 5.1 gives the details of the membership of the group during the simulation. Figure 5.2 shows that the permanent secret number of the user has been compromised.

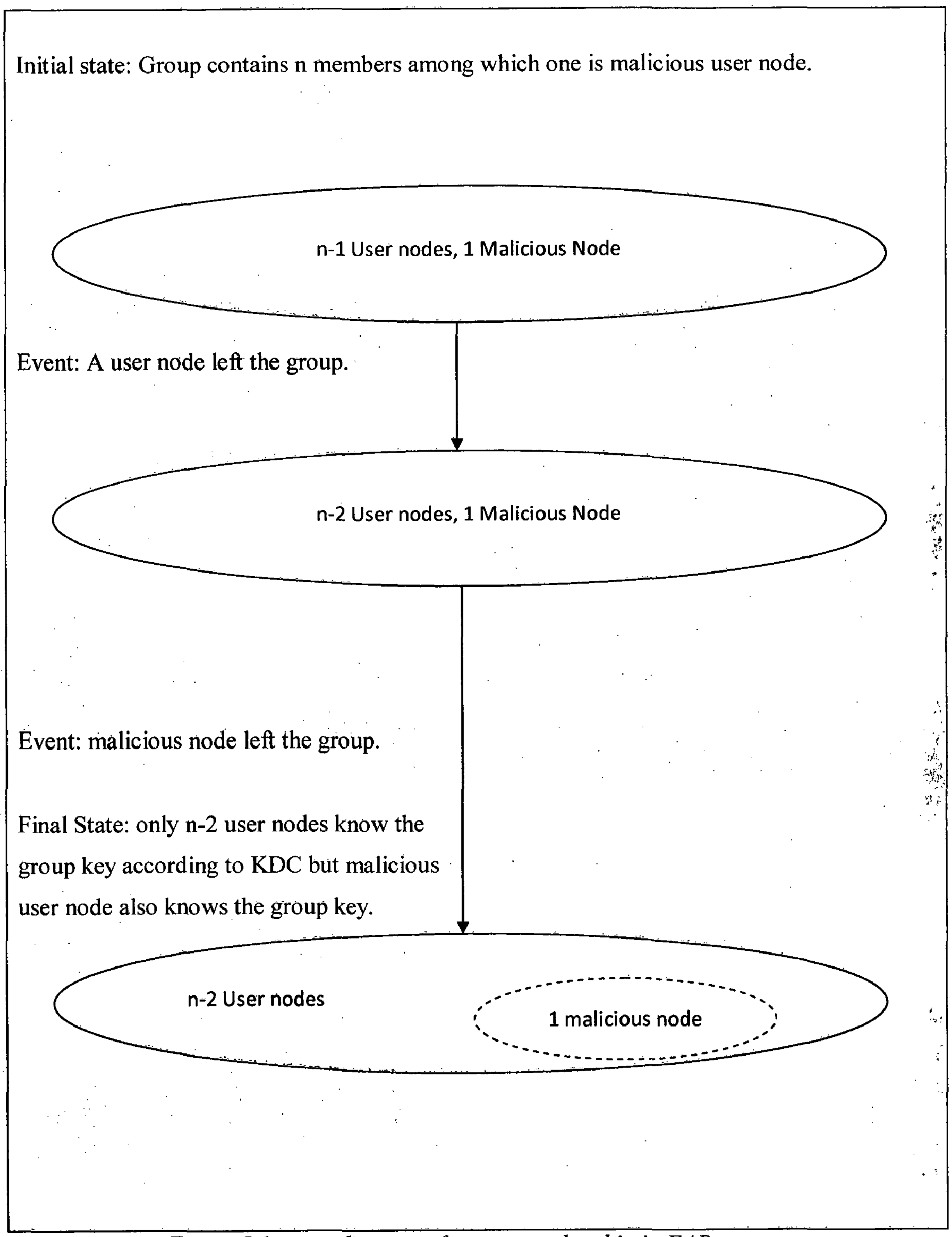
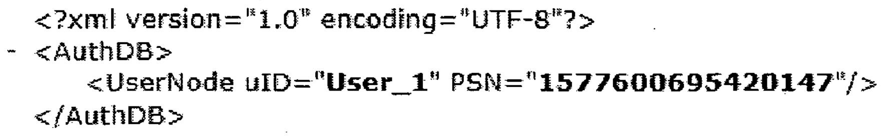
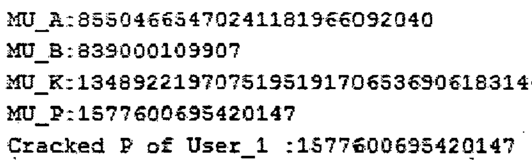

Initial state: Group contains n members among which one is malicious user node.
n-1 User nodes, 1 Malicious Node
Event: A user node left the group.
n-2 User nodes, 1 Malicious Node
Event: malicious node left the group.
Final State: only n-2 user nodes know the group key according to KDC but malicious user node also knows the group key.
n-2 User nodes
1 malicious node

Figure 5.1: state diagram of group membership in EAB



```
  <?xml version="1.0" encoding="UTF-8"?>
- <AuthDB>
      <UserNode uID="User_1" PSN="1577600695420147"/>
  </AuthDB>
```

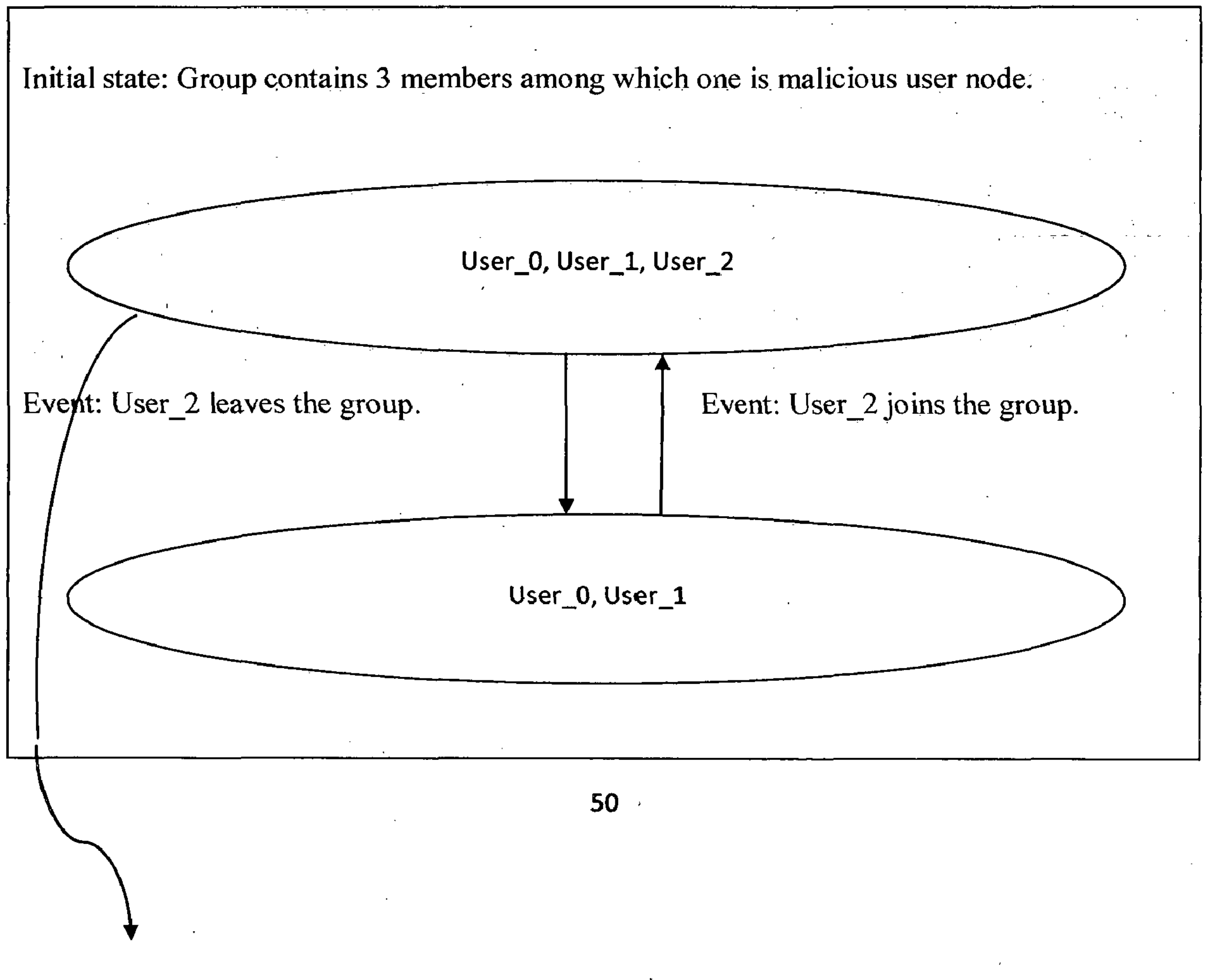


```
MU_A:8550466547024118196609204O
MU_B:839000109907
MU_K:134892219707519519170653690618314472439787
MU_P:1577600695420147
Cracked P of User_1 :1577600695420147
```

Figure 5.2: Screen shots showing the compromised user details match in EAB

5.1.2 Simulation Experiment 2(Cracking EABv2)

To make the understanding of the experiment simple, the experiment is conducted for group size of n = 3where user_0 being a user who joins the group and stays in the group till simulation ends. User_1 is the malicious user node that also joins the group and stays till simulation ends. UserNode User_2's behaviour is defined such that it repeatedly joins and leaves the group so that rekeying happens and enough data will be available for user_1 to compromise PSN_0.



Initial state: Group contains 3 members among which one is malicious user node.
User_0, User_1, User_2
Event: User_2 leaves the group.
Event: User_2 joins the group.
User_0, User_1

Event: User_2 leaves the group and User_1 finds out PSN_0

Final State: User_0's secret number is compromised

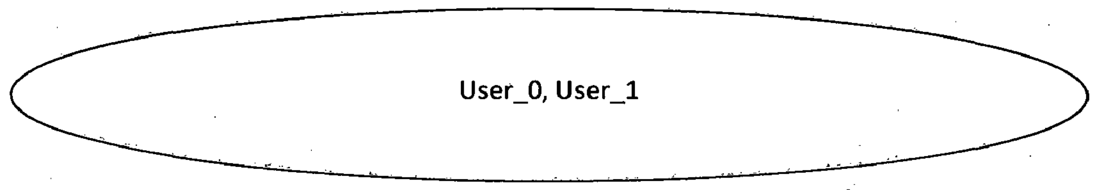
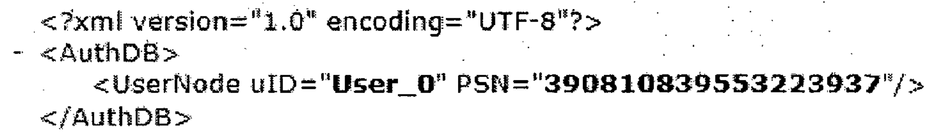
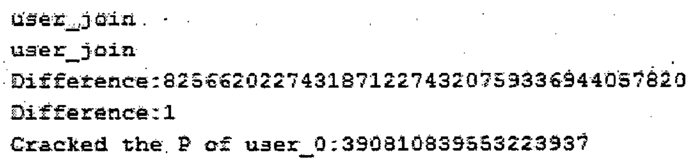


Figure 5.3: State diagram showing the group membership in EABv2



```
  <?xml version="1.0" encoding="UTF-8"?>
- <AuthDB>
     <UserNode uID="User_0" PSN="390810839553223937"/>
  </AuthDB>
```



```
user_join.
user_join
Difference:825662022743187122743207593369440578 20
Difference:1
Cracked the P of user_0:390810839553223937
```

Fig 5.4: screen shots showing the compromised user details match in EABv2

5.2 Performance Evaluation:

5.2.1 P-Leasel model

The number of users, service classes and the simulation mode has to be specified before the simulation starts. A group consisting of KDC and sub KDC is created and the number of membership events is equal to the number of users. Depending on the mode, one of join/leave membership event is initiated. The following graph shows the encryption/decryption on the system as a whole as the number of sub KDC in the group increases.

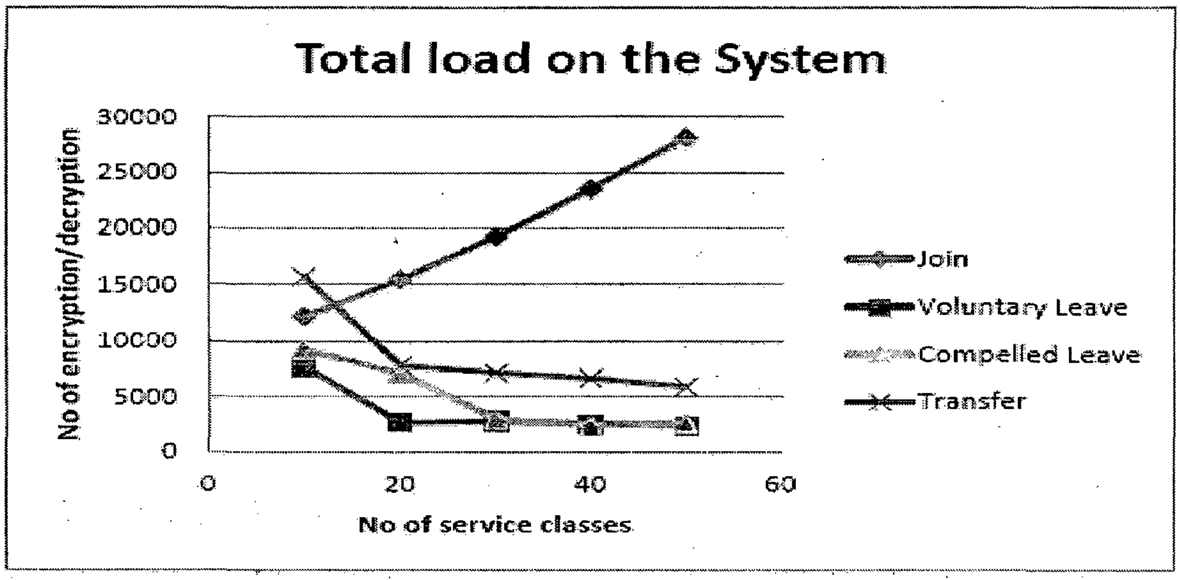


Figure5.5: Total overhead on the P-leasel system as the number of service classes increases

Results show that as number of service classes increase, overhead of member leaving the system is reduced. Figure 5.6 shows overheads on all components during the membership events specified.

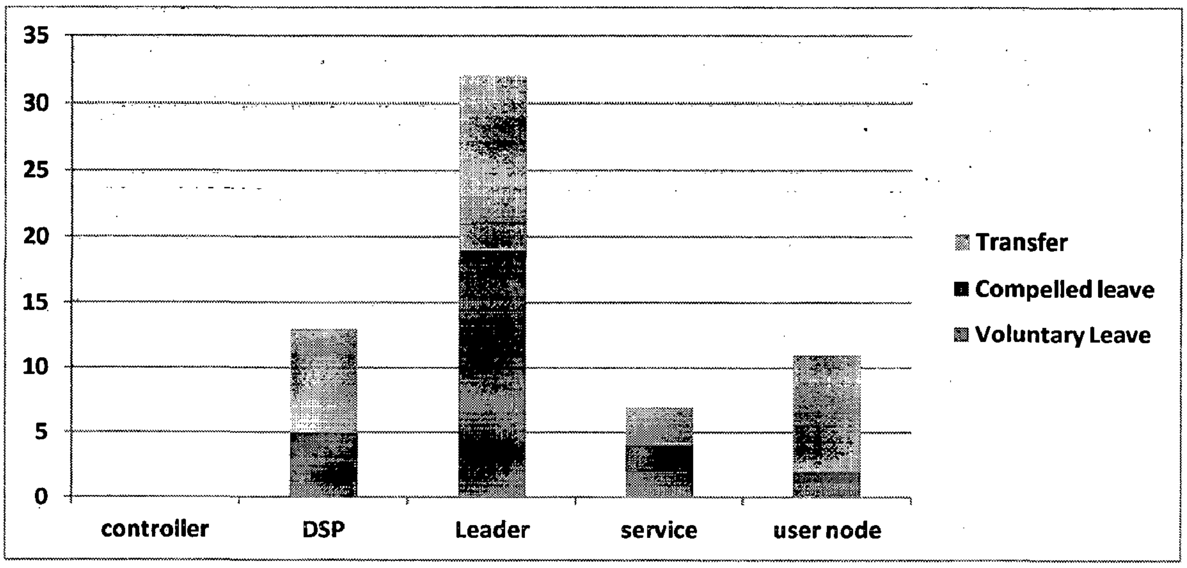


Figure 5.6: Computation load on different entities in P-Leasel

The figure shows that the computational load on leader nodes is high on the leader nodes that handle key distribution part. This means during rekeying a hign amount of overhead is due to using the encryption scheme.

The Register, Join and Leave protocols have been compared between the various schemes that are proposed and implemented. In each experiment, the group size is given as an input and the mode as to which protocol has to be simulated is specified. As, the simulation starts, a group is formed consisting of specified number of users and the specified membership event is simulated on the group. The computation overhead of various mechanisms during member leave is plotted in figure

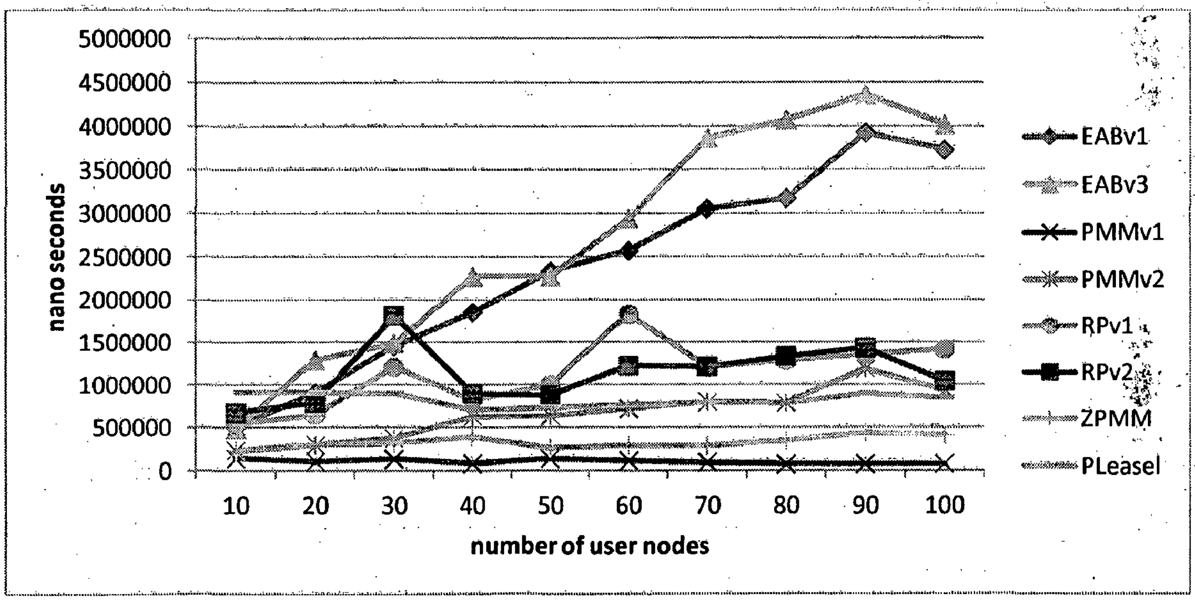


Figure 5.7: Comparison of computation overhead during leave among different methods

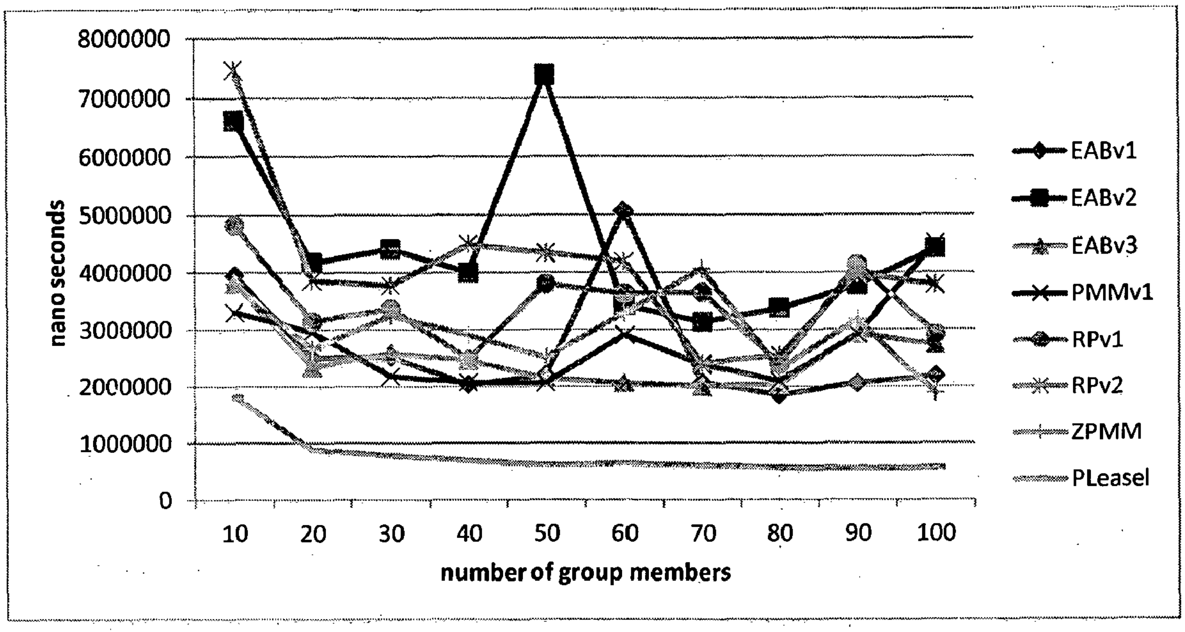


Figure 5.8: Comparision of computation overhead during leave among different methods

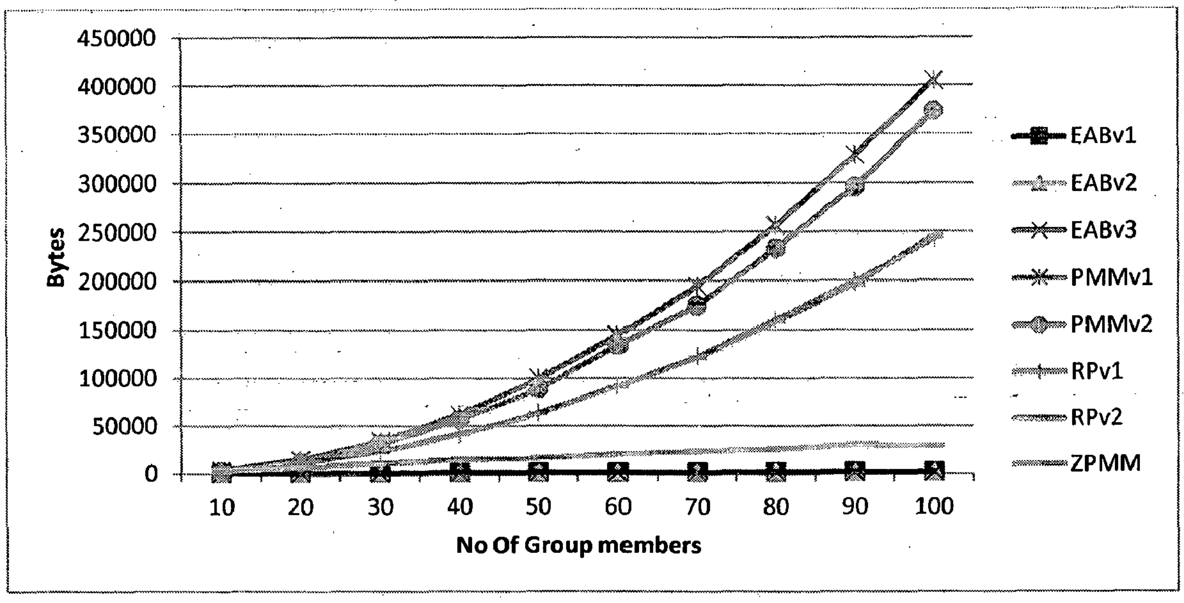


Figure 5.9: Comparision of bandwidth overhead during leave among different methods

The computation overhead during member join event is plotted in figure5.8. Similarly, the plot in figure 5.9 shows bandwidth consumption. The results show that the proposed schemes do not differ much in the over head during various membership events.

CHAPTER 6 CONCLUSIONS

In this dissertation, the need for a secure group communication mechanism that is scalable with efficient rekeying is discussed. Various existing solutions have been discussed P-Lease1 divides the group into subgroups to reduce communication overhead and also the frequency of rekeying. However, it uses encryption to distribute the group key to the members. So, when a member leaves the group the encryption overhead is proportional to the group size. The Prime Modulus method proposes an encoding scheme that needs a computation overhead independent of group size during both member join and leave. However, the size of the message to be sent to the group is proportional to group size. Hence, more bandwidth is needed. Moreover, the mechanism is not secure enough. Inorder to make it secure two prime method has been proposed which is also later found to be not secure enough. So, another method Rotational Prime Modulus method has been proposed. This is secure enough. Now, inorder to reduce the bandwidth group has been logically divided into zones and a method called Zone method is proposed which uses EAB method as a base. EAB method has been found out to not secure enough, so a modified scheme has been proposed EABv2. This is also found to be not secure enough. Further, a method EABv3 has been proposed. All the proposed methods have been implemented. Simulations have been made and the show that EAB, EABv1, PMM are not secure. The results also show that the proposed solutions pose a little overhead while security is provided.

REFERENCES

[1] I. Foster, C. Kesselman, and S. Tuecke."The anatomy of the grid: Enabling scalable virtual organizations. Int. J. High Perform. Comput. Appl., 15(3):200–222, 2001.

[2] Rajesh Ingle, G. Sivakumar, "EGSI: TGKA Based Security Architecture for Group Communication in Grid", 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing, IEEE Computer Society Melbourne, VIC, Australia , 17-20 May 2010, pp.34-42.

[3] Chen Lin, Huang Xiaoqin, Li Minglu, You Jinyuan," Secure Group Communication in Grid Computing", PDCAT 2004, LNCS 3320, 2004, pp. 604–607.

[4] N. Nagaratnam, P. Janson, J. Dayka, A. Nadalin, F. Siebenlist, V. Welch, I. Foster, S. Tuecke, "The Security Architecture for Open Grid Services," Open Grid Service Architecture Security Working Group, Global Grid Forum, 2002.

[5] Quan Zhou, Geng Yang, et al., "A Scalable SecurityArchitecture for Grid", Sixth International Conference on Parallel and Distributed Computing, Applications and Technologies, 2005, pp. 89 - 93.

[6] Grid Security Infrastructure (GSI), http://www.globus.org/security/

[7] Zou Xukai, Dai Yuan-Shun, Xiang Ran,"Dual-Level Key Management for secure grid communication in dynamic and hierarchical groups", Future Generation Computer Systems, In Press, Corrected Proof, Available online 22 December 2006.

[8] Chakrabarti. A., Damodaran, A., and Sengupta, S., "Grid Computing Security: A Taxonomy," IEEE Security and Privacy, 2007, pp. 44 – 51.

[9] P. Sakarindr ,N. Ansari,"Survey of security services on group communications", IET Inf. Secur., 2010, Vol. 4, Iss. 4, pp. 258–272.

[10] Dr Sudha Sadasivam G, Ruckmani V, Anitha Kumari K,"Secure Group Communication in Grid Environment", International Journal of Security (IJS), Volume (4), Issue (1), pp. 17 – 27, 2010.

[11] Rajesh Ingle, G. Sivakumar,"Tunable Group Key Agreement", 32nd IEEE Conference on Local Computer Networks, 2007, pp. 1017 – 1024.

[12] Chi-Chun Lo, Chun-Chieh Huang, Shu-Wen Chen,"An efficient and scalable EBS-based batch rekeying scheme for secure group communications", Military Communications Conference (MILCOM 2009), 2009, pp. 1 - 7.

[13] Yunfa Li, Hai Jin, Deqing Zou, Jieyun Chen, Zongfen Han,"A Scalable Service Scheme for Secure Group Communication in Grid", 31st Annual International Computer Software and Applications Conference (COMPSAC 2007), 2007, pp. 31 - 38.

[14] A.John Prakash, V.Rhymend Uthariaraj,"Multicrypt: A Provably Secure Encryption Scheme for Multicast Communication", First International Conference on Networks & Communications, 2009, pp. 246 - 253.

[15] Mary Vennila S, Sankaranarayanan V," P-LeaSel for GRID Environment", IJCSNS International Journal of Computer Science and Network Security, VOL.8 No.4, April 2008.

[16] P.Sivakami Priya, Dr.G.Sumathi,"An Improved Security and Trusting Model for Computational Grid", International Journal of Grid and Distributed Computing, 4(1), pp. 57 – 66, March 2011.

[17] Dr.M.Venkatesulu and Mr.K.Kartheeban, "EAB-Euclidian Algorithm Based Key Computation Protocol for Secure Group Communication in Dynamic Grid Environment", International Journal of Grid and Distributed Computing, IEEE, December 2010, pp. 45 – 53.

[18] Dr.M.Venkatesulu and Mr.K.Kartheeban, "An Efficient Certificate-Free Key Distribution Protocol for Secure Group Communication in Grid Environment", Journal of Computer Science, ISSN, 2011, 7(6):917-923.

[19] GridSim toolkit, www.buyya.com/gridsim

LIST OF PUBLICATIONS

[1] Sri Krishna chowdary Kankanala, Dr. Anjali Sardana, Dr. P. Sateesh Kumar,"An Extended Euclidian Algorithm Based Key Computation protocol for Secure Group Communication in Dynamic Grid Environment", 19th IEEE International Conference on High Performance Computing (HiPC), December 18-21, 2012, Pune, INDIA