SECURE KEY AGREEMENT AND AUTHENTICATION PROTOCOLS

B.Maheshwari, Assistant Professor, Dept. of Informatics,

Alluri Institute of Management Sciences,

Hunter Road, Warangal.

madhavigangapuram@gmail.com

ABSTRACT

We consider several distributed collaborative key agreement and authentication protocols for dynamic peer groups. There are several important characteristics which make this problem different from traditional secure group communication. They are:

1) Distributed nature in which there is no centralized key server;
2) Collaborative nature in which the group key is contributory (i.e., each group member will collaboratively contribute its part to the global group key); and
3) Dynamic nature in which existing members may leave the group while new members may join.

Instead of performing individual rekeying operations, i.e. recomputing the group key after every join or leave request, we discuss an interval-based approach of rekeying. We consider three interval-based distributed rekeying algorithms, or interval-based algorithms for short, for updating the group key: 1) the Rebuild algorithm; 2) the Batch algorithm; and 3) the Queue-batch algorithm. Performance of these three interval-based algorithms under different settings, such as different join and leave probabilities, is analyzed. We show that the interval-based algorithms significantly outperform the individual rekeying approach and that the Queue-batch algorithm performs the best among the three interval-based algorithms. More importantly, the Queue-batch algorithm can substantially reduce the computation and communication workload in a highly dynamic environment. We further enhance the interval-based algorithms in two aspects: authentication and implementation. Authentication focuses on the security improvement, while implementation realizes the interval-based algorithms in real network settings. Our work provides a fundamental understanding about establishing a group key via a distributed and collaborative approach for a dynamic peer group.

KEY WORDS

Authentication, dynamic peer groups, group key agreement, rekeying, secure group communication, security.

1. INTRODUCTION

As a result of the increased popularity of group-Oriented applications and protocols, group communication occurs in many different settings: from network layer multicasting to application layer tele-and video- conferencing. Regardless of the underlying environment, security services are necessary to provide communication *Authentication*, *Privacy* and *Integrity*.

Secure group communication is not a simple extension of secure two-party communication. Two-party communication can be viewed as a discrete phenomenon because it starts, lasts for a while and ends. Group communication is more complicated because it starts, and the group mutates (members leave and join) and there might not be a well defined end[2]. This results in necessity of providing security services for group communication among which KEY

AGREEMENT is the most important. The term key agreement means both the communicating persons and groups agree upon one common key called as secret key for secure communication between them.

The main aim of the paper is
To collaboratively generate a common key for peer to peer group communication.

1. To dynamically perform re-keying operation after batch of joins or leaves using Queue Batch algorithm and
2. To share resources using the generated group key.

With this main aim the system will provide the members of a group with secure common group key. This group key is generated collaboratively where in each node becomes a part of the key generation. The distributive nature avoids the usage of a centralized key server. The dynamic nature allows the members to leave and join the group and instead of performing individual rekeying operations the system uses Queue-batch algorithm for re-keying. The algorithm can substantially reduce the computation and communication workload in a highly dynamic environment. The group key is used for future communication among the members of the group.

1.2. Centralized Key server

For a secure group communication the System adopts a centralized approach, where there is a central coordinator who is responsible for the group key computation. So the centralized system completely depends upon the single coordinator for the group communication and rekeying is also performed for individual join and leave operations. The system mainly involves the Centralized Key server where in the entire systems depends on centralized key server for

- Key Generation
- Key Distribution and
- Rekeying.

One way to achieve secure group communications is to have a symmetric key, called ***group key***, and shared only by group members (also called users). The group key is distributed by a key server which provides group key management service. Messages sent by a member to the group are encrypted with the group key, so that only members of the group can decrypt and read the messages. If a user wants to join the group, the user sends a join request to the key server. The user and key server mutually authenticate each other using some protocol. If authenticated and accepted into the group, the user shares with the key server a symmetric key, called the user's ***individual key*****.**

For a group of N users', initially distributing the group key to all users requires N messages each encrypted with an individual key (the computation and communication costs are proportional to group size N). To prevent a new user from reading past communications (called backward access control) and a departed user from reading future communications (called forward access control), the key server may ***rekey*** (change the group key) whenever group membership changes. For large groups, join and leave requests can happen frequently. Thus, a group key management service should be scalable with respect to frequent key changes.

Key Distribution Centre

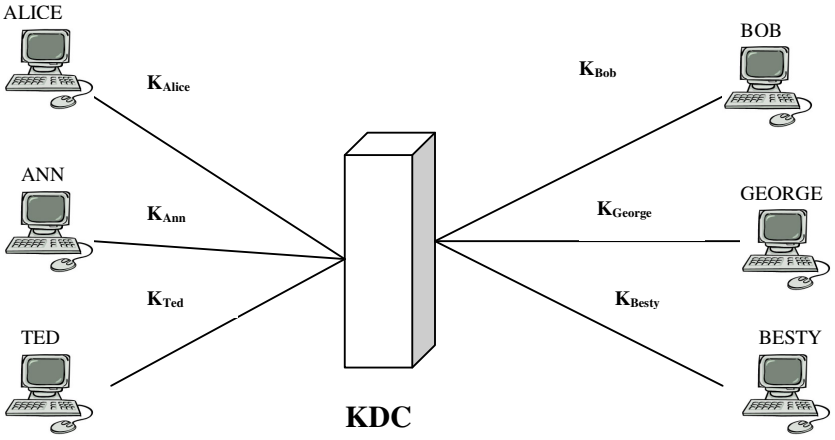
The centralized key server is also called as Key Distribution Centre KDC in some contexts. To support group communication each KDC maintains information of users like every

user's individual key which the user shares with the KDC. In this centralized approach if any 2 users say A & B wish to communicate then both have to share their respective individual keys with KDC which will help for the purpose of Authentication in further communication. If user A wants to send any message to user B then user A has to first contact the KDC. The KDC then authenticates the user with his Individual key and the generates a Session Key and sends back to user A. This Session Key will be used by user A to communicate with user B.

Communication using Centralized Key Server

In the centralized approach we make use of a "Trusted Third Party" which is also called as "Key Server" or "KDC – Key Distribution Centre". Each user of the group has to establish a secret key called the users ***individual key*** with the Key Server.

This scenario can be shown as below for a group of 6 users

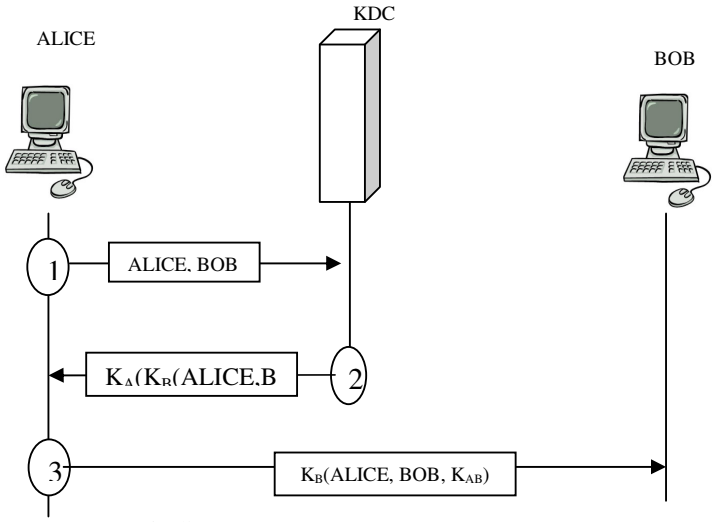


A secret key is established between the KDC and each member of the group. The secret key of Alice with the KDC is referred to as K_{Alice} and similarly the other keys like KBob , K_{George} and so on. Now if Alice wants to send a confidential message to Bob then the process will be

1. Alice sends a request to the KDC stating that she needs to communicate with bob and send some confidential data
2. The KDC informs to Bob about Alice's request
3. If Bob agrees then a *Session Key* is created between Alice and Bob for the purpose of communication.

Session key Generation by the KDC (or Key Server)

The process of *Session Key* generation between Alice and Bob can be shown as follows



K_A - Secret Key of Alice
K_B - Secret Key of Bob
K_{AB} - Session Key

Limitations of the KDC:

In this approach when number of users are less the job of KDC to maintain

1. All the users individual keys
2. Generation of session keys for communication
3. Distribution of session keys

are considerably possible with no problem. But as the number of users increase then it becomes an overload for the KDC to maintain the data as well as generation and distribution of session keys. This arises the problem of Single Point Failure

1.2. Rekeying

When the communication is between static groups then the Group key generated once can be used through out the communication. But when we have dynamic groups where members of the group join or leave the group frequently then the group key generated once will not serve the purpose. This is because to support

- backward access control i.e. to prevent a new user from reading past communications , and
- forward access control i.e. to prevent a departed user from reading future communications

The group key must be newly generated when ever any new user joins the group or any existing user leaves the group. This method is called as Individual Rekeying

The *key graph* approach has been proposed for scalable rekeying. In this approach, besides the group key and its individual key, each user is given several *auxiliary keys (or intermediate*

keys). These auxiliary keys are used to facilitate rekeying. *Key graph* is a data structure that models user-key and key-key relationships. *Key tree* is an important type of key graph where key-key relationships are modeled as a tree. All the users of the group who are part of communication are arranged in a form of a Tree with the leaves representing the actual group members holding their individual keys. The intermediate non leaf nodes can be called the Auxiliary keys or sub keys or intermediate keys. The key at the root node will be the actual Group key which the group members use for their communication.

Limitations of rekeying

It is easier to rekey after a join than a leave. After a join, the new group key can be sent via unicast to the new member (encrypted with its individual key) and via multicast to existing members (encrypted with the previous group key). After a leave, however, since the previous group key cannot be used, the new group key may be securely distributed by encrypting it with individual keys. This straightforward approach, however, is not scalable. In particular, rekeying costs 2 encryptions for a join and N-1 encryptions for a leave, where N is current group size.

Individual rekeying, however, has two problems.

- First, it is relatively inefficient.
- Second, there is an out-of-sync problem between keys and data.

Hence the major drawbacks of the centralized key server system can be listed as below:

- Key information depends on centralized key server.
- All users must agree upon the key generated by the Key Server only without any choice
- This will be a problem if the Key Server (the trusted third party) itself is not trust worthy.
- Since key information depends on centralized key server there are more chances for problem of single point failure
- Computational and Communication cost is more.
- Individual re-keying is done. Whenever a member joins or leaves.
- More resources are used for re-keying because it is done for each join or leave operations.

2. DISTRIBUTIVE COLLABORATIVE DYNAMIC SYSTEM

- Distributive nature means there is *no centralized key server*
- Collaborative nature means in which the group key is *contributory* (i.e., each group member will collaboratively contribute its part to the global group key).
- Dynamic nature in which existing members may leave the group while new members may join

To provide privacy in group communication, it is important that members of the group can establish a common secret key for encrypting group communication data. To illustrate the utility of this type of applications, consider a group of people in a peer-to-peer network having a closed and confidential meeting. Since they do not have a previously agreed upon common secret key, communication between group members is susceptible to eavesdropping. To solve the problem, we need a *secure distributed group key agreement and authentication protocol* so that people can establish and authenticate a common group key for secure and private communication. Note that this type of key agreement protocols is both distributed and contributory in nature: each member of the group contributes its part to the overall group key [5].

2.1. Group Key Generation in a Contributory Approach

In the centralized system we had only one key being used by each user which is called his individual key. This system can be called as the Symmetric key cryptosystem. This approach makes use of the Public Key Cryptosystem also called as the Asymmetric key cryptosystem. Here each user has a set of keys , one key is called as the **Public** Key which is made public to all the other users and the other key is called as the **Private key** which will be the secret key of the user which only the user holds.

Diffie-Hellman Key Exchange Algorithm

In the special case of a communication group having only two members, these members can generate a group key using the Diffie–Hellman key exchange protocol. This is a public key cryptographic system which means every user will have sets of keys one called the Private Key and the other called as the Public Key or Blinded Key. The process can be explained as:

- all users agree on global parameters:
  -- Large prime integer or polynomial *P*
  -- An integer α that is a primitive root of ***P***

- **α** is called as the primitive root of **P** if power's of **α** modulo **P** generate all integers from 1 to P – 1 i.e. $\boldsymbol{\alpha \bmod P}$, $\boldsymbol{\alpha^2 \bmod P}$, $\boldsymbol{\alpha^3 \bmod P}$, ……. , $\boldsymbol{\alpha^{p-1} \bmod P}$ are all distinct and consist of all integers from 1 to P – 1.
- each user U perform the following operations:

  Chooses a Private Key (number): $X_U < p$

  Computes Public Key as: $Y_U = \alpha^{XU} \bmod p$

  Y_U is send to the other communicating party.

- Every user computes a common secret key using his own private key and the opposite users shared public key which provides authentication for the communication.
- When the secret key generated by both the users is same then they both authenticate one another and starts communication.

2.2. Tree Based Group Diffie-Hellman Protocol

To efficiently maintain the group key in a dynamic peer group with more than two members, we use the tree-based group Diffie–Hellman (TGDH) protocol .Each member maintains a set of keys, which are arranged in a hierarchical binary tree. We assign a node id **V** to every tree node. For a given node, **V** we associate a secret (or private) key $\mathbf{K_v}$ and a blinded (or public) key $\mathbf{BK_v}$. All arithmetic operations are performed in a cyclic group of prime order with the generator.

Therefore, the blinded key of node can be generated by

$$\mathbf{BK_v = \alpha^{Kv} \bmod p}$$

Where p is any large prime number and α is a primitive root of p.

Each leaf node in the tree corresponds to the individual secret and blinded keys of a group member Mi. Every member holds all the secret keys along its key path starting from its associated leaf node up to the root node. Therefore, the secret key held by the root node is shared by all the members and is regarded as the group key.

The figure below illustrates a possible key tree with six members M1 to M6. For example, member M1 holds the keys at nodes 7, 3, 1, and 0. The secret key at node 0 is the group key of this peer group.

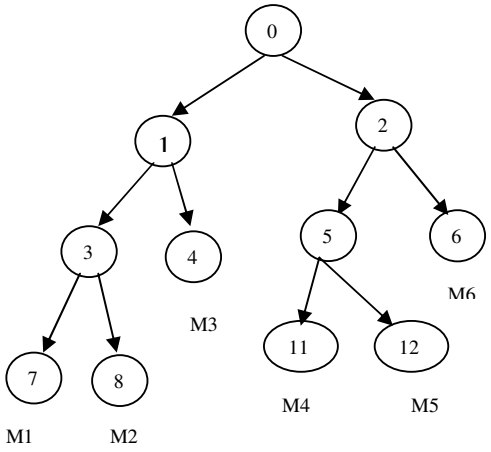


Figure 2.2A: A Key Tree with 6 members

The node ID of the root node is set to 0. Each nonleaf node consists of two child nodes whose node ID's are given by 2v+1 and 2v+2 . Based on the Diffie–Hellman protocol, the secret key of a nonleaf node can be generated by the secret key of one child node of **v** and the blinded key of another child node of v. Mathematically, we have

$$K_V = (BK_{2V+1})^{K2V+2} \bmod P$$
$$= (BK_{2V+2})^{K2V+1} \bmod P$$
$$= \alpha^{K2V+1\ K2V+2} \bmod P$$

Unlike the keys at nonleaf nodes, the secret key at a leaf node is selected by its corresponding group member through a secure pseudo random number generator. Since the blinded keys are publicly known, every member can compute the keys along its key path to the root node based on its individual secret key.

Example:

To illustrate, consider the key tree in above figure. Every member Mi generates its own secret key and all the secret keys along the path to the root node. For example, member M1 generates the secret key K_7 and it can request the blinded key BK_8 from M2, BK_4 from M3, and BK_2 from either M4 , M5,or M6.

- Given M1's secret key K_7 and the blinded key BK_8, M1 can generate the secret key K_3 according to the above given formula as:

Let the values of global public parameters be

P = 5 and α = 2

Let $K_7 = 3$ and $\mathbf{B}K_8 = 2$

Then $K_3 = (BK_8)^{K_7} \bmod P = 2^3 \bmod 5 = 8 \bmod 5 = 3$

- Given the blinded key BK_4 and the newly generated secret key K_3 , M1 can generate the secret key K_1 based on given formula as:

We have $K_3 = 3$ and let $\mathbf{B}K_4 = 6$

Then $K_1 = (BK_4)^{K_3} \bmod P = 6^3 \bmod 5 = 216 \bmod 5 = 1$

Given the secret key K_1 and the blinded key BK_2 , M1 can generate the secret key K_0 at the root.

We have $K_1 = 1$ and let $\mathbf{B}K_2 = 7$

Then $K_0 = (BK_2)^{K_1} \bmod P = 7^1 \bmod 5 = 7 \bmod 5 = 2$

- From that point onwards, any communication in the group can be encrypted based on the secret key (or group key) $\mathbf{K_0}$.

To provide both backward confidentiality (i.e., joined members cannot access previous communication data) and forward confidentiality (i.e., left members cannot access future communication data), rekeying, which means renewing the keys associated with the nodes of the key tree, is performed when-ever there is any group membership change including any new member joining or any existing member leaving the group.

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

Consider the Tree of 6 users as shown above in figure 7

Following figures depicts a MEMBER LEAVE EVENT

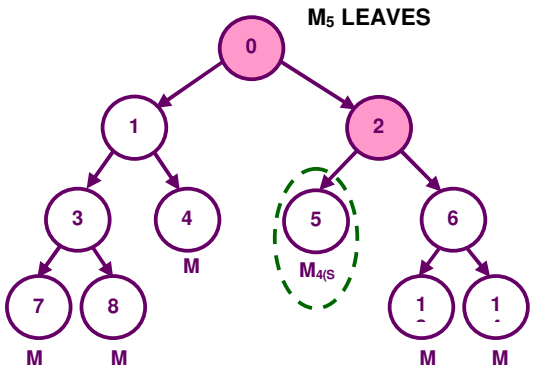


Figure 2.2B: Member LEAVE Operation on a Key Tree

- M_4 becomes the ***Sponsor***. It rekeys the secret keys K_2 and K_0 and broadcasts the blinded key BK_2.
- M_1, M_2 and M_3 compute K_0 given BK_2.
- M_6 and M_7 compute K_2 and then K_0 given BK_5

Following figures depicts a MEMBER JOIN EVENT

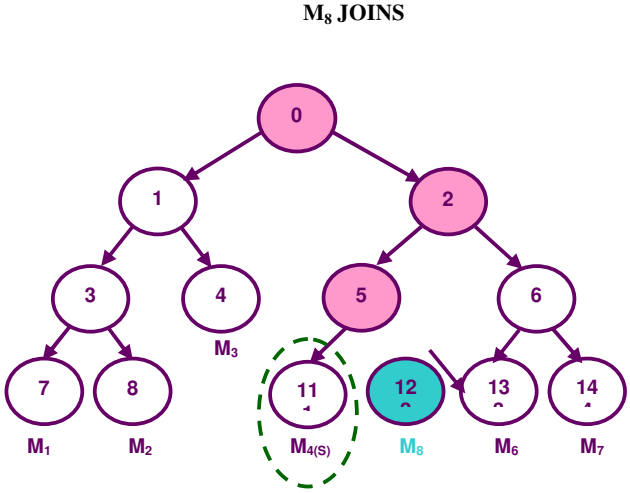


Figure 2.2C: Member JOIN Operation on a Key Tree

- M8 broadcasts its individual blinded key BK12 on joining.
- M4 becomes the sponsor again. It rekeys K5, K2 and K0 and broadcasts the blinded keys BK5 and BK2.
- Now everyone can compute the new group key.

2.3. Interval Based Rekeying or Batch Rekeying

To address the two problems of individual rekeying as explained in section 2.4, we propose the use of periodic *batch rekeying or Interval Based Rekeying*. In batch rekeying, the key server waits for a period of time, called a ***rekey interval***, collects the entire join and leave requests during the interval, generates new keys, constructs a rekey message and multicasts the rekey message. This Batch Rekeying can over come the problems of individual rekeying

We consider three *interval-based distributed rekeying algorithms* (or *interval-based algorithms* for short) called as the

- Rebuild algorithm
- Batch algorithm, and
- Queue-batch algorithm.

The first 2 approaches of **Rebuild** and **Batch algorithm** perform all rekeying steps at the beginning of every rekeying interval. This results in high processing load during the update instance and there by delays the start of the secure group communication. Thus, more effective algorithm is proposed which we call the ***Queue-batch*** **algorithm**

This approach uses distributed environment and has its own main advantages over centralized method.

1. The system does not depend upon a single coordinator to find the group key. So the single point failure will not cause serious damage to the whole system.
2. Each member in the group is autonomous in nature; hence the Group key is arrived from the contribution of all the legitimate members in the group.
3. since the system does not depends upon a single coordinator and hence the level of Security has been increased.
4. More over the system performs rekeying on a batch or group of join and leave operations. This will overcome the problems faced by the individual rekeying method

2.3.1. The Rebuild Algorithm

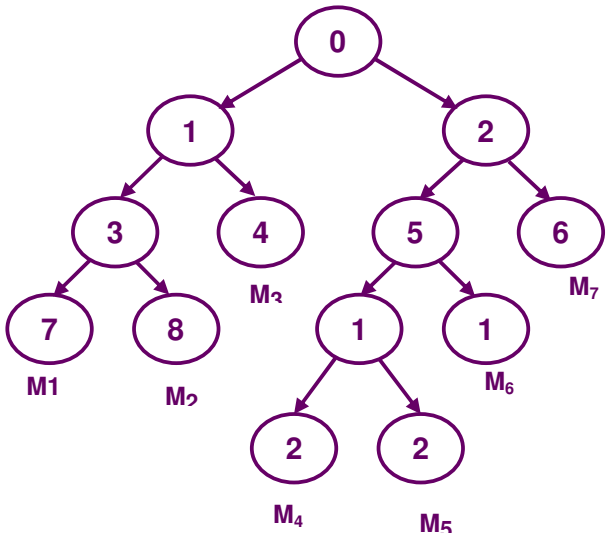
The motivation of the Rebuild algorithm is to *minimize* the resulting tree height so that the rekeying operations for each group member can be reduced. At the beginning of every rekeying interval, we reconstruct the whole key tree with all existing members that remain in the communication group, together with the newly joining members. The resulting tree is a *left-complete* tree, in which the depths of the leaf nodes differ by at most one and those deeper leaf nodes are located at the leftmost positions.

The Pseudo code of the Rebuild Algorithm can be given as

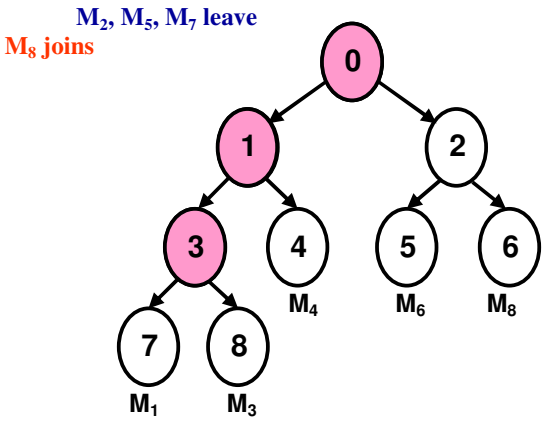
Rebuild *(T, $M^{j,}$ J, M^{l}, L)*

1. Obtain all members from T and store in M'
2. Remove the L leaving members in Ml^{j} from M'
3. Add the J new members in M^{l} to M'
4. create a new binary tree T' based on members in M' and set T = T'
5. Elect all members to be sponsors
6. Rekey renewed nodes and broadcast new blinded keys in T

Let us consider the following key tree with 7 members to understand the interval based rekeying algorithms



Applying the Rebuild algorithm the resulting key tree will be



Key Tree after applying REBUILD Algorithm

The key tree consists of five members and has all non leaf nodes renewed. Besides, the sponsors include all the five members. Rebuild is suitable for some cases, such as when the membership events are so frequent that we can directly reconstruct the whole key tree for simplicity, or when some members lose the rekeying information and the simplest way of recovery is to rebuild the key tree.

2.3.2. The Batch Algorithm

The Batch algorithm is based on the centralized approach which is now applied to a distributed system without a centralized key server. The pseudo-code of the Batch algorithm is given as follows

BATCH(T , M^j , J , M^l , L)

1. if (L = = 0) /* Pure Join Case * /
2. { create a new Tree **T'** based on new members in $\mathbf{M}^{j}$;
3. either add **T'** to the shallowest node of **T** such that the merge will not increase the height of the result tree , or add **T'** to the root node of **T** if the merge to any node of T will increase the tree height ;
4.} else /* L > 0 */
5.{ sort M^{l} in the ascending order of the associated node IDs of the members and store the results in $M^{l,S}$ = { $M_1^{l,S}$, $M_2^{l,S}$, …… , $M_L^{l,S}$ };
6. if (L > J)
7. if (J > 0) {
8. replace the departed nodes of { $M_1^{l,S}$, $M_2^{l,S}$, …… , $M_L^{l,S}$ } with J joined nodes
9. }
10 remove remaining L – J leaving leaf nodes and promote their siblings ;
11 . } else / * J >= L */
12. { divide M^{j} in to L subgroups G = < G_1, G_2 , …… , G_L > such that the first J mod L subgroups contain [J/L]+1 new members and the rest contain [J/L] new members ;
13. create L sub trees < T'_1 , T'_2 , …… , T'_L > for the sub groups G ;
14.replace the departed nodes of { $M_1^{l,S}$, $M_2^{l,S}$, …… , $M_{J \bmod L}^{l,S}$ }with the roots of < T'_1 , T'_2 , …… , $T'_{J \bmod L}$ > and the remaining departed nodes with the roots of remaining sub trees ;
15.}
16.}
17. elect the members to be sponsors if they are new members , or the rightmost members of the sub trees rooted at the siblings of the departed nodes or the replaced nodes in T;
18. if (Sponsor) / * responsibility of the sponsor */
19. rekey the renewed nodes and broadcast the new blinded keys ;

2.3.3. The Queue - Batch Algorithm

The Queue-batch algorithm is divided into two phases, namely the ***Queue-sub tree phase*** and the ***Queue-merge*** phase.

The first phase occurs whenever a new member joins the communication group during the idle rekeying interval. In this case, we append this new member in a temporary key tree.

The second phase occurs at the beginning of every rekeying interval and we merge the temporary tree (which contains all newly joining members) to the existing key tree. The pseudo-codes of the Queue-sub tree phase and the Queue-merge phase are illustrated as follows.

QUEUE – SUBTREE (T')

1. if(a new member joins) {
2. if (T' = = NULL) /* no new member in T' */
3. create a new tree T' with the only new member;
4. else { /* there are new members in T' */
5. find the insertion node ;
6. add the new member to T' ;
7. elect the right most member under the sub tree rooted at the sibling of the joining node to be the sponsor ;
8. if(sponsor) /* sponsors responsibility */
9. rekey renewed nodes and broadcast new blinded keys ;
10. }
11. }

QUEUE – MERGE(T , T' , M^{l} , L)

1. if(L = = 0) { /* there is no leave */
2. add T' to either the shallowest node of T' such that the merge will not increase

the resulting tree height , or the root node of T if the merge to any locations will increase the resulting height ;
3. } else { /* there are laves */
4. add T' to the highest leave position of the key tree T ;
5. remove remaining L – 1 leaving leaf nodes and promote their siblings
6. }
7. elect members to be sponsors if they are the right most members of the sub tree rooted at the sibling nodes of the departed leaf nodes in T , or they are the rightmost member of T' ;
8. if(sponsors) /* sponsors responsibility */
9. rekey renewed nodes and broadcast new blinded keys ;

3. CONCLUSION

We consider several distributed collaborative key agreement protocols for dynamic peer groups. The key agreement setting is performed in which there is no centralized key server to maintain or distribute the group key. We show that one can use the TGDH protocol to achieve such distributive and collaborative key agreement. Rekeying is performed when ever any new user joins the Group or any existing user leaves the group. This is done to achieve Forward Confidentiality i.e. a user who left the group cannot access the data any more and Backward Confidentiality which means a user newly joining the group will not know the previously accessed data.

To reduce the rekeying complexity, we propose to use an interval-based approach to carry out rekeying for multiple join and leave requests at the same time, with a tradeoff between security and performance. In particular, we show that the Queue-batch algorithm can significantly reduce both computation and communication costs when there exist highly frequent membership events. We also address both authentication and implementation for the interval-based key agreement algorithms. As with other applications, there is certainly a scope for improvement in this application too. New modules may be included to increase the compatibility of the project. Once these improvements have been done, the majority of the features that make an application more excellent can be achieved.

REFERENCES

1. M. Steiner, G. Tsudik, and M. Waidner, "Key agreement in dynamic peer groups," *IEEE Trans. Parallel Distrib. Syst.*, vol. 11, no. 8, pp. 769–780, Aug. 2000.
2. W. Diffie and M. Hellman, "New directions in cryptography," *IEEE Trans. Inf. Theory*, vol. 22, no. 6, pp. 644–654, 1976.
3. "Tree-based group key agreement," *ACM Trans. Inf. Syst. Security*, vol. 7, no. 1, pp. 60–96, Feb. 2004.
4. Security Services for Dynamic Peer Groups An IBM Research Paper.
5. "Batch Rekeying for Secure Group Communications" Xiaozhou Steve Li, Yang Richard Yang, Mohamed G. Gouda, Simon S. Lam Department of Computer Sciences University of Texas at Austin Austin, TX 78712-1188
6. Distributed Collaborative Key Agreement and Authentication Protocols for dynamic Peer Groups Patrick P. C. Lee, John C. S. Lui*, Senior Member, IEEE*, and David K. Y. Yau*, Member, IEEE*
7. "A block-free TGDH key agreement protocol for secure group communications" by Xukai Zou Department of Computer and Information Sciences Indiana University - Purdue University Indianapolis, USA , Byrav Ramamurthy Department of Computer Science and Engineering University of Nebraska-Lincoln, USA
8. Cryptography and Network Security by Behrouz A. Forouzan . TMH chapter 15 (page 438 – 440)

9. Cryptography and Network Security – Principles and Practices by WILLIAM STALLINGS Chapter 10: Key Management
10. S. Blake-Wilson and A. Menezes, "Authenticated Diffie-Hellman key agreement protocols," in *Proc. 5th Annu. Workshop on Selected Areas in Cryptography (SAC'98)*, 1998, vol. LNCS 1556, pp. 339–361.
11. A. Perrig, "Efficient collaborative key management protocols for secure autonomous group communication," in *Int. Workshop on Cryptographic Techniques and E-Commerce (CrypTEC '99)*, Jul. 1999, pp. 192–202.
12. C. K.Wong, M. Gouda, and S. S. Lam, "Secure group communications using key graphs," *IEEE/ACM Trans. Netw.*, vol. 8, no. 1, pp. 16–30, Feb. 2000.

AUTHOR

B Maheshwari M.Sc (CS), M.Tech (CSE), working as an assistant professor in Alluri Institute of Management Sciences, Hunter Road, Warangal, since 2006.