Hierarchical Based Group Key Transfer for Secure Group Communication

Kanimozhi Sakthivel[1], Velumadhava Rajasekaran[2], Selvamani Kadirvelu[2], Kannan Arputharaj[1]
[1]Department of Information Science and Technology, Anna University, India
[2]Department of Computer Science and Engineering, Anna University, India

Abstract: *In this research paper, a scalable and efficient multicast secure protocol for large and dynamic systems that relies on trusted Key Generation Center (KGC) is proposed. In this protocol, we partition the entire group into several subgroups which is controlled by Intermediate Controller (IC). Our protocol is based on Iolus protocol and the hierarchical structure of LKH protocol. The decomposition of subgroup is organized in an hierarchical manner as it is in the LKH protocol which helps us to reduce the complexity for a member to join or leave from O(n) to O(log m), where n represents the number of entire group members and m represents the number of each subgroup members. Out protocol performance is compared with that of Iolus and LKH protocol. The performance is enhanced especially when there is a member leave operation.*

Keywords: *Group communication, decentralized protocols, distributed protocols, logical key hierarchy and secure multicast.*

Received April 6, 2012; accepted March 13, 2013; published online October 29, 2015

1. Introduction

The key distribution is one of the most important issues for providing secure group communication. Multicasting can be defined as the process of sending a message to selected group members [1, 24]. Many web based internet applications such as stock quotes, online games etc., can benefit from secure multicast communication. For example, consider a stock data distribution group, which distributes stock information to a group of users around the world. Only the authorized members who have subscribed to the service should get the stock data information. Member who joins newly to the group should receive stock information immediately, but should not receive the stock information that has been shared prior to member joining the group. Also, when the member leaves the group, they should not receive any further stock information.

In most of these applications, members of the group typically receive identical information from a single or multiple senders. Hence, providing a common session encryption key to all the group members will reduce the number of message units to be encrypted by the senders. The group key should be changed frequently whenever member join or leave the system to achieve a high level of security by means of re-key operation [4, 5, 6, 9, 22, 23]. The secrecy should be maintained such that a former group member has no access to the current communication and a new member has no access to previous communications.

Message confidentiality and message authentication becomes a major problem in secure multicast [8, 10, 17, 18, 27]. In order to solve this problem a symmetric key, which is a secret group key must be distributed to all the multicast group members.

These members will have the authority to access the multicast data using the group key. Moreover, there are many protocols that have been proposed to solve the problem of group key distribution. These protocols can be classified into three categories. Centralized, decentralized and distributed key management approaches [3, 5, 6, 16, 22, 23, 27]:

- Centralized Approach: A single entity is employed for controlling the entire group and hence a group key management protocol seeks to minimize storage requirements, computational power on both client and server sides and bandwidth utilization.
- Decentralized Approach: The management of a large group is divided among subgroup managers, trying to minimize the problem of concentrating the work in a single place.
- Distributed Approach: There is no explicit entity like Key Generation Center (KGC) and the members themselves do the key generation. All members can perform access control and the generation of the key can be either contributory, meaning that all member contribute some information to generate the group key, or done by one the members.

In this paper, we propose a new protocol to efficiently manage a shared group key among group members. The key can be used to encrypt the transmitted data for group communication confidentiality. The proposed key management protocol relies on a distributed and decentralized key server that coordinates protocol runs to distribute the group key to group members securely. In this protocol, we divide the entire group into several subgroups. The complexity for member join or

member leave is reduced from $O(n)$ to $O(m)$ and further each sub group is organized in a hierarchical manner as in the LKH protocol and thus subgroup complexity is reduced from $O(m)$ to $O(log\ m)$. The group key is shared only to the Intermediate Controller (IC). The members belongs to the particular subgroup must know the subgroup key to decrypt the data. The subgroup key is obtained using member secrets and server secrets assigned to each group member and also the inverse value of the member secret helps us to manage the subgroup key when there is a member leave operation. Each member in the group will have to store the inverse value of other group members except his own secret value. Whenever member leaves the group, the KGC server just sends the identity of the member to other subgroup members.

The centralized approaches [11, 16, 23, 26] are based on the concept of LKH protocol where a Key Distribution Center (KDC) or Key distribution unit maintains a key tree. The Group Controller (GC) acts as a root and shares a group key with other group members and each member know the keys from its leaf to the root. For example, consider a hierarchical tree with the known set of keys $\{K_{2,1}, K_{1,1}, K_1\}$ for u_1 where K_1 represents the group key. In order to, maintain the backward secrecy, if u_8 joins the group as shown in Figure 1. The keys $\{K_{2,4}, K_{1,2}, K_1\}$ has to be changed into $\{K'_{2,4}, K'_{1,2}, K'_1\}$. The KGC server encrypts the new key with that of the old key and sends the multicast message as: $\{K'_{2,4}\}K'_{2,4}, \{K'_{1,2}\}K'_{1,2}, \{K'_1\}K$,. u_7 can obtain $K'_{2,4}$ by decrypting the first part of the message using $K_{2,4}$. u_5, u_6 and u_7 can obtain $K'_{1,2}$ by decrypting the second part using $K_{1,2}$ and members from u_1 to u_7 can obtain K'_1 by decrypting the third part using K_G, and the message sent by uncast is: $\{K'_{2,4}, K'_{1,2}, K'_1\}K_8$. u_8 can easily obtain the new keys by decrypting the message using K_8. Similarly, if u_8 leaves the group as shown in Figure 2, the key server has to change $\{K'_{2,4}, K'_{1,2}, K'_1\}$ to maintain forward secrecy. Since, the leaving member knows the old group key and the old intermediate keys, the changes has to be made with all group members by KGC server. In the decentralized approaches [1, 10, 15, 17, 18, 19, 21, 27] the entire group is divided into small subgroups. In order to overcome the single point of failure problem by centralized approaches, different controller acts as a intermediate server to manage each subgroup.

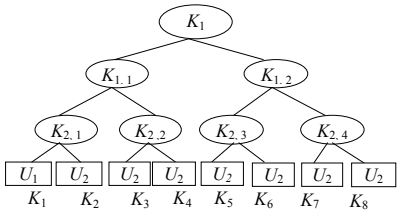

Figure 1. Member (u_8) joins communication system.

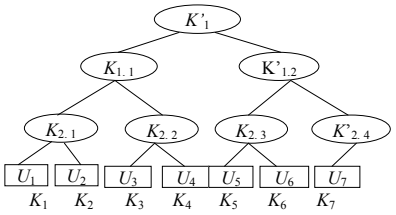

Figure 2. Member (u_8) leaves communication system.

If the failure occurs in one IC will not affect the entire group. Mitrra proposed Iolus [15], which is a framework with hierarchical agents that helps to divide the entire group into different subgroups. Group Security Agent (GSA) maintains the subgroup member. The individual group key has been used in this approach such that if there is any changes in one subgroup will not affect the other subgroup.

In [12] the principle used is secure session key management with key recovery function. Most of the distributed protocol approach is based on the DH key agreement protocol. More works of this approach can be found in [2, 7, 14, 20, 25]. The DH exchange is used to establish a symmetric key between two entities. However, DH approach suffers from the scalability problem as it can generate session only for two entities. The remaining paper is organized as follows. Section 2 explores the proposed work and the implementation details. Section 3 analyzes and discusses the results obtained from the work. Section 4 concludes the proposed and implemented work and suggested some possible enhancements.

2. The Proposed Work

Based on the above survey in this secure multicast communication, it is necessary to propose a new model to solve the identified issues. The main objective is to establish a symmetric key between all group members in order to provide the security in group communication. In case of a change occurs in the group membership by joining or leaving the group, the group key should be updated to maintain backward secrecy and forward secrecy.

The proposed protocol is based on the idea of splitting up the entire group into small and several subgroups. Our protocol is based on Iolus and LKH protocol. The decomposition of subgroups reduces the complexity for a member join or leave from $O(n)$ to $O(m)$, where n represents the number of member in the entire group and m represents the number of members in the subgroup. Each subgroup is further divided into hierarchical tree structure which is based on LKH protocol that reduces the complexity of member join or leave from $O(m)$ to $O(log\ m)$. The proposed protocol works in a hierarchy of two level controllers; the first is the GC and the second is the IC. The GC shares the group key with IC's to transmit the data in a secure manner to their subgroup members. Each IC works as the server for their subgroup.

The hierarchical structure for the subgroup in the protocol is shown in Figures 3 and 4. K_{SG1} represents the key for the subgroup 1. The following assumptions are made for our proposed protocol. They are:

- ICs are trustable entities.
- ICs are static entities means they can only join the group but they cannot leave the group.
- Members are organized in a hierarchy of a binary tree to enhance the key distribution at leave operation.
- The total number of subgroups is n, the height of each subgroup is h, so each subgroup contains m=2h users.
- In each subgroup, the balance of the key distribution is considered for grouping operation.
- When a new member joins the group, the GC is responsible for allocating new member in a appropriate subgroup. GC will search for an empty place in all subgroups, if it does not find any empty places in the subgroup, then a new subgroup is created to allocate the new member.

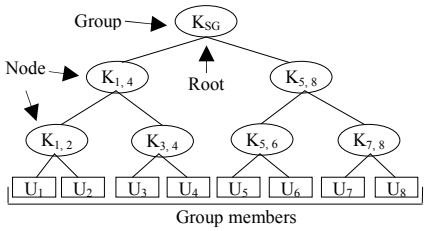

Figure 3. Structure of subgroup hierarchy.

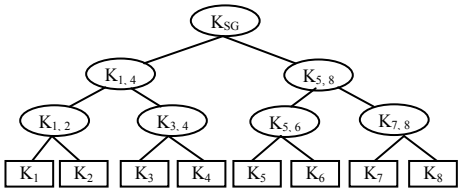

Figure 4. Proposed subgroup hierarchy.

From Figure 3, we can notice that the number of the subgroup members is m=8 and the height h=$\log_2$8= 3:

- u_1 and u_2 agree on the node key $K_{1,2}$. u_3 and u_4 agree on the node key $K_{3,4}$. u_5 and u_6 agree on the node key $K_{5,6}$. u_7 and u_8 agree on the node key $K_{7,8}$.
- u_1, u_2 and u_3, u_4 agree on the node key $K_{1,4}$. u_5, u_6 and u_7, u_8 agree on the node key $K_{5,8}$.
- u_1, u_2, u_3, u_4 and u_5, u_6, u_7, u_8 agree on the sub-group key K_G.

In the proposed protocol, the group key is known only by the ICs. The ICs shares the symmetric key with the members in the subgroup and this shared group key becomes the subgroup key. The main work of the ICs is to decrypt the data coming to their subgroups using that group key and then re-encrypt the data by their subgroup keys. The re-encrypted data is sent to their associated subgroup members by multicast. There is no need for a group member to know the group key instead they can know only the subgroup key to decrypt the data. The operations of our proposed protocol are explained as follows:

2.1. Key Structure

In this system, each user U_i is required to share a random secret value S_i and also a permanent secret id P_i is assigned for each member U_i in the subgroup. Using those secrets the subgroup key for the group members is calculated by Equations 1 and 2.

$$M_j=((P_1 \oplus S_1)*(P_2 \oplus S_2), ...,* (P_n \oplus S_n))+K_j \text{ for } j= 1, 2, ..., n \quad (1)$$

$$K_{SG1}= M_1 \bmod (P_i \oplus S_i) \text{ for all } i \quad (2)$$

Where M_j is the message of the subgroup which is used to calculate the subgroup key and K_j is the secret key for that subgroup. The node keys are constructed by multiplying the components of its two child node keys in the logical key tree. This algorithm for rekeying can be illustrated using a simple example of a multicast subgroup of seven members:

- Member's u_1 and u_2 own keys K_1 and K_2 respectively, node keys $K_{1,2}$ and $K_{1,4}$, and the subgroup key K_{SG1}.
- Members u_3 and u_4 own keys K_3 and K_4 respectively, node keys $K_{3,4}$ and $K_{1,4}$, and the group key K_{SG1}.
- Members u_5 and u_6 own keys K_5 and K_6 respectively, node keys $K_{5,6}$ and $K_{5,7}$, and the group key K_{SG1}.
- Member u_7 own keys K_7, the node key $K_{5,7}$ and the group key K_G. The keys are calculated as follows:

a. *Level* 1:

$$K_1= M_1 \bmod (P_1 \oplus S_1)$$
$$K_2= M_1 \bmod (P_2 \oplus S_2)$$
$$K_3= M_1 \bmod (P_3 \oplus S_3)$$
$$K_4= M_1 \bmod (P_4 \oplus S_4)$$
$$K_5= M_1 \bmod (P_5 \oplus S_5)$$
$$K_6= M_1 \bmod (P_6 \oplus S_6)$$
$$K_7= M_1 \bmod (P_7 \oplus S_7)$$

b. *Level* 2:

$$K_{1,2}= M_1 \bmod (P_1 \oplus S_1)(P_2 \oplus S_2)$$
$$K_{3,4}= M_1 \bmod (P_3 \oplus S_3)(P_4 \oplus S_4)$$
$$K_{5,6}= M_1 \bmod (P_5 \oplus S_5)(P_6 \oplus S_6)$$

c. *Level* 3:

$$K_{1,4}= M_1 \bmod (P_1 \oplus S_1)(P_2 \oplus S_2)(P_3 \oplus S_3)(P_4 \oplus S_4)$$
$$K_{5,6}= M_1 \bmod (P_5 \oplus S_5)(P_6 \oplus S_6)(P_7 \oplus S_7)$$

When the GC receives the partial keys from its two child node keys he puts his secret value K and generates the subgroup key K_{SG1} using Equation 3 and multicasts that key to all subgroup members.

$$K_{SG1} = M_1 \ mod \ (P_i \oplus S_i) \ for \ all \ i = 1, 2, 3, ..., n \quad (3)$$

2.2. IC Join

In the case of IC join, suppose that a IC_{n+1} wants to join the group, the new group key has to be distributed to the new *IC* i.e., group key is changed from $K(G)$ into $K'(G)$. The GC distribute the new group key to the existing IC by broadcasting the message as; $\{K'(G)\}K(G)$, $\{K'(G)\}K(GC, IC_{n+1})$, where $K'(G)$ is the new group key, $K(G)$ is the old group key and $K(GC, IC_{n+1})$ is the symmetric key shared between the GC and the new IC. To obtain the new group key, the existing ICs will decrypt the first part of the message using the old group key and the newly joined IC decrypts the second part of the message using the symmetric key shared with him. Therefore, once a new IC joined, only there is a update of the group key in ICs, it will not affect other group members. The GC is responsible for two key encryptions; one for the old *IC*s and the other for the new IC. Suppose for example, a sender wants to send a message to subgroup number 1. The sender encrypts his message $\{M\}$ by using a key K_M and that key is also encrypted by using the group key K_G. The sender sends the following message to the IC1: $\{M\}K_M, \{K_M\}K_G$. When the IC_1 received that message he uses the group key (K_G) to decrypt the second part of the message and get the K_M which he uses to decrypt the first part of the message to get the origin message and then encrypts it using the subgroup1 key (K_{IC1}) and multicast the ciphered message to his subgroup members.

2.3. Member Join

Whenever member wants to join a group, he sends a "join" request to the KGC server which acts as a GC. The GC will direct the new member to the specified group. The IC of that subgroup takes the join request from the GC and to maintain the backward secrecy of that particular subgroup changes are made. Suppose for example, u8 wants to join the group, he sends a "join" request to the GC. The GC will search for the correct subgroup in our case subgroup 1 (SG1) and direct the new member to SG1. The subgroup hierarchy when member joins is shown in Figure 5. The following operations are performed by SG1.

- Creates a new node key $K_{7,8}$. K_7 becomes its left child and the key of the new member K_8 becomes its right child.
- Assigns a identity P_8 value to the new member and its inverse value P_8^{-1} is calculated. This inverse value is multicast to all the existing subgroup members.
- The subgroup key is changed to maintain the backward secrecy.
- When a new member join or leave then the path keys are updated.
- The new group key and the inverse values of the existing members are distributed to the newly joined member via unicast.
- The group key and the inverse value of the new member is distributed to all subgroup members via multicast.

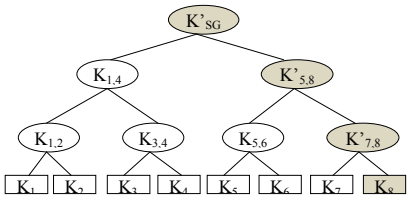

Figure 5. Subgroup 1 hierarchy when u8 joins.

The IC1 sends the following message to the members in his subgroup:

$$\{K'_{SG1}, (P_8 \oplus S_8)^{-1}\} \ K_{SG1}, \{K_{5,8}\} \ K_{5,7}, \ \{K_{7,8}\}K_7, \{K'_{SG1}, K_{5,8}, K_{7,8}, (P_1 \oplus S_1)^{-1}, (P_2 \oplus S_2)^{-1}, ..., (P_7 \oplus S_7)^{-1}\}K_8$$

To obtain the new subgroup key K'_{SG1} and to obtain the inverse value of the new member secret first part of the message is decrypted using the old group key K_{SG1}. The members u_5, u_6 and u_7 will able to obtain the new node key $K_{5,8}$ by decrypting the second part of the message using the old node key $K_{5,7}$ and obtains the node key $K_{7,8}$ by decrypting the third part of the message. u_8 obtain his path keys and the inverse values of the other member's secrets by decrypting the last part of the message.

$$K_{7,8} = M_1 \ mod \ (P_7 \oplus S_7) \ (P_8 \oplus S_8)$$

$$K_{5,8} = M_1 \ mod \ (P_5 \oplus S_5) \ (P_6 \oplus S_6) \ (P_7 \oplus S_7) \ (P_8 \oplus S_8)$$

$$K'_{SG1} = M'_1 mod \ (P_1 \oplus S_1) * (P_2 \oplus S_2), ..., * (P_8 \oplus S_8)) + K'_1$$

2.4. Member Leave

When a member of the subgroup wants to leave, he sends a "leave" request to the IC of his subgroup. The IC sends the identity of the member who is leaving the group. Also, the IC sends the keys which should be updated to the remaining subgroup members. When the subgroup members receive the message from IC_1, they will use the inverse value of the member who leaves the group to update the keys. Suppose for example, if u_8 wants to leave the subgroup SG1, the keys K_{SG1}, $K_{5,8}$, $K_{7,8}$ should be updated to maintain forward secrecy. According to our protocol, there is no need to send the updated keys to the remaining members. Instead, the IC_1 just prepares one message for indicating u_8 leaves and the keys must be updated. The IC_1 sends the message $\{u_8, K'_{SG1}\}K'_{SG1}$, $\{K_{5,8}\}K_{5,8}$. In this message the keys in the message are encrypted using the same keys. By seeing this type of message, the remaining members identify that there is a leave operation and the leaving member identity is 8. They use the inverse value of $u_8(P_8 \oplus S_8)^{-1}$ to update the sent keys.

$$M' = ((P_1 \oplus S_1) * (P_2 \oplus S_2), \ldots, *(P_{j-1} \oplus S_{j-1})) \quad ((P_{j+1} \oplus S_{j+1}) *, \ldots, (P_n \oplus S_n)) + K' \tag{4}$$

$$K'_{SG1} = M' mod (P_i \oplus S_i) for\ i = 1, 2, \ldots, 7 \tag{5}$$

$$K'_{5,8} = M' mod (P_5 \oplus S_5)\ (P_6 \oplus S_6)\ (P_7 \oplus S_7)) \tag{6}$$

3. Protocol Analysis

We analyse and compare the performance of the proposed protocol with that of Iolus and LKH based protocol. The following assumptions are made for analyzing the join and leave operation of the above mentioned protocols:

- The total number of each subgroup members is *m* members. Therefore, the total number of the entire group is *n*=number of subgroups *N*×*m* members.
- The height *h* of the proposed protocol is $log_2 m$, i.e., the total number of group members in each subgroup is equal to $m=2^h$.

The Proposed work is implemented in java with a minimum group size of 16 and maximum group size of 4096. Based on the implementation results are obtained.

3.1. Key Generation

Key generation overhead is defined as the number of keys generated for the member join and leave takes place by the key server and a member node.

From Table 1, it is observed that for LKH protocol the complexity for generating the session when a new member joins is $log_2\ n$ and this is to maintain the backward secrecy. Also when the member leaves the group the keys are generated with the complexity of $\log_2$ n [13]. Similarly, it is observed for Iolus protocol that the complexity for generating the group key while join and leave operation is totally 2 keys. Since the Iolus protocol is based on Decentralized approach i.e. new member joins the IC instead of GC, the subgroup key generated is 1 for join and 1 for leave. The generated subgroup key is distributed to all subgroup members [16]. In our proposed protocol, it is noticed that the complexity for generation of subgroup keys when member join in the subgroup is $\log_2$ m along with the session key and the secret value of the joining member. Whenever there is a leave operation, the IC will not generate any keys. Instead the identity of the leaving member is multicast to the entire subgroup members. Figures 6 and 7 shows the key generation for join and leave operation with different group size.

Table 1. Key generation for join and leave operation.

Protocols	Key Server		Member Nodes	
	Join	Leave	Join	Leave
LKH	O($\log_2$ n)	O($\log_2$ n)	0	0
Iolus	O(1)	O(1)	0	0
Proposed Protocol	O($\log_2$ m)	0	0	0

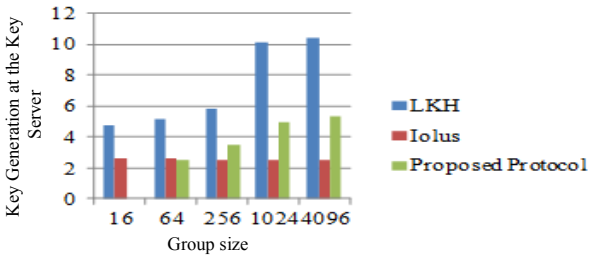

Figure 6. Number of key generation at key server for join.

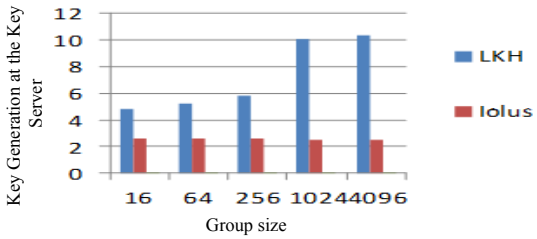

Figure 7. Number of key generation at key server for leave.

By analyzing the complexity of LKH, Iolus and our protocol, the key generation overhead is reduced in our protocol compared to LKH protocol. In LKH the key server generates $log_2\ n$ keys whereas our protocol generates $log_2 m$ keys.

3.2. Encryption/Decryption

Encryption/Decryption overhead can be defined as the number of encryptions at the key server and the number of decryptions for a member node.

In the LKH protocol, whenever member joins a group, the root key server performs $3log_2n$ encryptions and for member leave, the key server performs $2log_2n$ encryptions. Similarly, for member join and leave operation, each member in the LKH protocol makes $\log_2 n$ decryptions [14]. In the Iolus protocol, the key server makes two encryptions at a member join operation. At a member leave, the key server performs *m*-1 encryptions. Whenever there is members join and leave operations, each member in this protocol performs only one decryption [15]. The proposed protocol is entirely different from the LKH protocol in the number of encryptions. It uses a one-way function tree, the key server makes one encryption per node but in the LKH protocol it makes two encryptions. During a join operation, the key server must update the path keys of the member who joins the subgroup.

The key server encrypts the new keys by its corresponding old keys. Since, each subgroup of *m* members is organized in a hierarchical tree, the key server makes *log*2*m* encryptions. In order to obtain the new keys, the key server update its path keys encrypted by its individual key and send it to the joining member such that the key server makes other *log*2*m* encryptions for the joining member. So, the overall encryption performed by the key server at the join operation is equal to 2*log*2*m*. When a member leaves a subgroup, the key server must update the path keys of the leaving member so it performs *log*2*m* encryptions. When the

members of a subgroup receive the encrypted $log2m$ keys from the key server, each member makes $log2m$ decryptions in a member join operations, but it doesn't make any decryptions at a member leave operation as it only factors the leaving member's secret value from the necessary keys by using the inverse value of the leaving member.

Table 2 shows time complexity of encryption /decryption overhead at key server and number of decryptions at a member node at both join and leave operations. The proposed protocol has the smallest number of encryptions and decryptions at the key server and member nodes respectively at the leave operation as shown from Table 2.

Table 2. Encryption/Decryption for join and leave operation.

Protocols	Join	Leave
LKH	$O(\log_2 n)$	$O(\log_2 n)$
Iolus	O(1)	O(m)
Proposed Protocol	$O(\log_2 m)$	O(1)

From Figures 8 and 9, it is observed that proposed protocol is more efficient when it is compared with LKH protocol in terms of encryption/ decryption of messages.

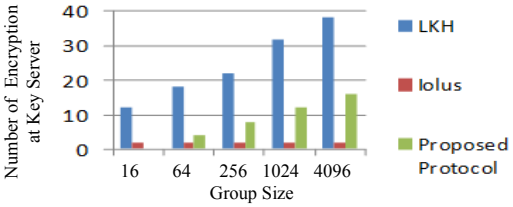

Figure 8. Encryption at the key server for join.

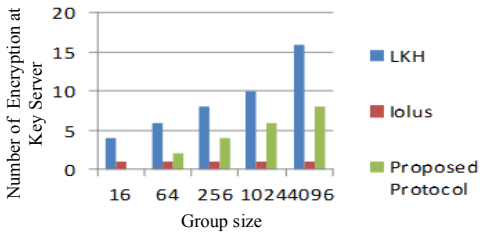

Figure 9. Decryption at the member node for join.

3.3. Multicast Message

Message size can be defined as the number of keys included in one multicast message. Table 3 shows the message size for join and leave operation.

Table 3. Message size for join and leave operation.

Protocol	Join	Leave
LKH	$O(\log_2 n)$	$O(\log_2 n)$
Iolus	O(1)	O(m)
Proposed Protocol	$O(\log_2 m)$	O(1)

From the analysis of LKH protocol, whenever the member join the total number of keys to be multicast is $2log_2 n$-1 keys and for member leave the message size to be multicast is $2log_2 n$ keys. While in Iolus protocol, for member join the multicast message includes only 2 keys and for leave it include m-1 keys [16]. In the proposed protocol, the message contains $2log_2m$+1 keys at a join operation and new path keys of the joining member and the inverse value of the joining member is all encrypted by the old subgroup key. At a leave operation, the sub GC sends a multicast message including only the identity of the leaving member to all the subgroup members. Figure 10 shows the number of message sizes at the join operation with respect to group size.

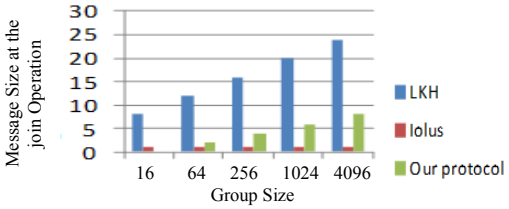

Figure 10. Multicast message sizes for member join

3.4. Storage Complexity

Storage Complexity can be defined as the number of keys stored in the key server and by the member node. Table 4 shows the storage complexity at key server and at member node.

Table 4. Storage complexity.

Protocols	Key Server	Member Node
LKH	O(n)	$O(\log_2 n)$
Iolus	O(m)	O(1)
Proposed Protocol	O(m)	O(m)

In LKH protocol, the total number of keys that will be stored is $2n$ based on the d-ary protocol using the formula $[d/d\text{-}1]n$ where n is the number of group members. Each member has to store its path keys from itself to the root node along with the session key. So, each member has to store $log_2 n$+1 key [14]. While in Iolus protocol, it requires an m individual key that has to be stored by key server and by the member node. In the proposed protocol, the key storage in the key server is the sum of $2m$-1 node keys, m inverse values of the group member secret. So, the key server stores $3m$ keys. Each member has to stores its path keys $log2m$ and m-1 inverse values of the group member secret. So, each member stores m+log2m-1.

4. Experimental Setup

The experimental setup consists of group of 10 members. Members 1, 2, ..., 10 has the permanent prime id (Prime) as, $P1$=55837, $P2$=55603, $P3$=35353, $P4$=54709, $P5$=60799, $P6$=45953, $P7$=40847, $P8$=39461, $P9$=42709, P10=58909 respectively along with the group key K=22971. Moreover, the secret key shared by the users are $S1$=28931, $S2$=37123, $S3$=12347, $S4$=13745, $S5$=16231, $S6$=31234, $S7$=21467, $S8$=25431, $S9$=17237, $S10$=21719.

By applying Equation 1 Message is generated by KGC and the generated messages are M=551311

57819910041417986790291061886647009280000 along with the group key K=22971 is manipulated according to Equation 1 and then it is broadcasted to all the ten members in the hierarchical group. At level1 k_1 is computed as $M\%(P_1 \oplus S_1)$. i.e., K_1= 55131157819910041417986790291061886647009302971% (55387 ⊕ 28931). The value obtained from the above calculated results in the original key value which is 22971. Similarly all the other members in the group calculate their keys by doing $M\%(P_i \oplus S_i)$, i=2, 3, ..., 10. At level-2 $k_{1,2}$ is computed as M%$(P_1 \oplus S_1)$ $(P_2 \oplus S_2)$. i.e., $K_{1,2}$= 55131157819910041417986790291061886647009302971% (55387 ⊕ 28931)(55603 ⊕ 37123).

The value obtained from the calculated results in the original value which is 22971. Similarly all the other member of pair $K_{3,4}$, $K_{5,6}$ etc., are calculated.

We have taken K (Key) sizes as 64,128, 512, 1024bits and the value of S (prime) has been taken has 64, 128, 512 and 1024bits. When a non-group member m_k attempts to compute the group key with a value P_k=43651, S_k=45079 and the above message M. The member mk will not able to calculate the key. i.e., $M\%(P_k \oplus S_k)$=55131157819910041417986790291061866470093029 71%(43651 ⊕ 45079)=2559≠22971≠K.

5. Conclusions

In this paper, we proposed an efficient and a scalable protocol for secure group communication to solve the issues such us message size, storage complexity, and other overheads in distributing the symmetric key among the group members. The decomposition of this technique reduces the complexity for a member join and leave from $O(n)$ to $O(log2\ m)$, where n represents the entire group member and m represent the subgroup members. The performance of the proposed protocol is compared with that of the LKH protocol and Iolus protocol. The comparison is undertaken according to the complexity for key generation, multicast message size, storage complexity overhead and based on encryption/decryption overhead. The results show that the proposed protocol enhances the group performance in terms computation overhead specifically at the time of member leave operation. The main drawback in this approach is that whenever there is a member join or leave operation a inverse value is calculated which is distributed to all the subgroup members. Hence, when the number of subgroup members increased the storage of messages in each member node increase as well as there is a need to store the inverse value of other members in the group. Hence, the storage complexity in each member node increases. Further works in this direction could be implemented and applied in cloud environment. This approach will reduce the storage complexity and increase the scalability and the system performance.

References

[1] Ahamad M., "Multicast Communication in Distributed Systems," *IEEE Computer Society Press Technology Series*, pp. 43-55, 1990.

[2] Aslan H., "A Scalable and Distributed Multicast Security Protocol Using a Subgroup-Key Hierarchy," *Elsevier Computers and Security*, vol. 23, pp. 320-329, 2004.

[3] Bonmariage N. and Leduc G., "A Survey of Optimal Network Congestion Control for Unicast and Multicast," *International Journal of Network Security*, vol. 12, no. 2, pp. 61-74, 2011.

[4] Cao J., Liao L. and Wang G., "Scalable Key Management For Secure Multicast Communication In The Mobile Environment," *Pervasive and Mobile Computing*, vol. 2, no. 2, pp. 187-203, 2006.

[5] Challal Y. and Seba H., "Group Key Management Protocols: A Novel Taxonomy," *International Journal of Information Technology*, vol. 2, no. 1, pp. 105-118, 2005.

[6] Chan K. and Chan S., "Key Management Approaches to Offer Data Confidentiality For Secure Multicast," *IEEE Network*, vol. 17, no. 5, pp. 30-39, 2003.

[7] Chen X., MA B. and Yang C., "M-CLIQUES: Modified CLIQUES Key Agreement for Secure Multicast," *Elsevier Computers and Security*, vol. 26, pp.238-245, 2007.

[8] Chua T. and Pheanis D., "Bandwidth Conserving Multicast VOIP Teleconference System," *International Journal of Network Security*, vol. 7, no. 1, pp. 42-48, 2008.

[9] Gharout S., Yacine C., and Bouabdallah A., "Scalable Delay-Constrained Multicast Group Key Management," *International Journal of Network Security*, Vol. 7, No.2, pp. 160-174, Sep.2008.

[10] Harte L., "Introduction to Data Multicasting, IP Multicast Streaming for Audio and Video Media Distribution," *Althos Publishing*, 2008.

[11] Jun Z., Yu Z., Fanyuan Ma., Dawu Gu., and Yingcai B, "An Extension Of Secure Group Communication Using Key Graph," *Elsevier Information Sciences*, vol. 176, pp. 3060-3078, 2006.

[12] Kanyamee K. and Sathitwiriyawong C., "High-availability decentralized cryptographic multi-agent key recovery," *International Arab Journal of Information Technology*, vol. 11, no. 1, pp. 52-58, 2014.

[13] KeiWong C., Gouda M., and Lam S., "Secure Group Communications Using Key Graphs," *IEEE/ACM Transactions On Networking*, vol. 28, no. 4, pp. 16-30, 2000.

[14] Kim Y., Perrig A., and Tsudik G., "Communication Efficient Group Key

Agreement," *IFIP SEC'01 Conference*, pp. 229-244, France, 2001.
[15] Mittra S., "Iolus: A Framework For Scalable Secure Multicasting," *ACM SIGCOMM Computer Communication Review*, vol. 27, no. 4, pp. 277-288, 1997.
[16] Ng W., Howarth. M., Sun Z. and Cruickshank H., "Dynamic Balanced Key Tree Management for Secure Multicast Communications," *IEEE Transactions on Computers,* vol. 56, no. 5, pp. 590-605, 2007.
[17] Paul S., "Multicasting on the Internet and Its Applications," *Springer-Verlag*, 1998.
[18] Peterson L. and Davie B., *Computer Networks: A Systems Approach*, Morgan Kaufmann Publishers Inc., Fourth Edition, 2007.
[19] Peyravian M., Matyas S., and Zunic N., "Decentralized Group Key Management For Secure Multicast Communications," *Computer Communications*, vol. 22, pp. 1183-1187, 1999.
[20] Pour A., Kumekawa K., Kato T. and Itoh S., "A Hierarchical Group Key Management Scheme for Secure Multicast Increasing Efficiency of Key Distribution in Leave Operation," *Computer Networks*, vol. 51, pp. 4727-4743, 2007.
[21] Rafaeli S. and Hutchinson D., "A Survey of Key Management for Secure Group Communication," *ACM Computing Surveys,* vol. 35, no. 3, pp. 309-329, 2003.
[22] Rafaeli S. and Hutchinson D., "Hydra: A Decentralized Group Key Management," *Proceedings of the 11th IEEE International WETICE: Enterprise Security Workshop*, pp. 62-67, June 2002.
[23] Setia S., Zhu S., and Jajodia S., "A Scalable and Reliable Key Distribution Protocol For Multicast Group Rekeying," *Technical report, George Mason University*, 2002.
[24] Srinivasan R., Vaidehi V., Rajaraman R., and Kanagaraj S., "Secure Group Key Management Scheme For Multicast Networks," *International Journal of Network Security*, vol. 11, no. 1, pp. 33-38, 2010.
[25] Steiner M., Tsudik G., and Waidner M., "Key Agreement In Dynamic Peer Groups," *IEEE Transactions on Parallel and Distributed Systems,* vol. 11, no. 8, pp. 769-780, 2000.
[26] Wallner D., Harder E., and Agee R., "Key Management for Multicast: Issues and Architectures," *National Security Agency*, RFC 2627, 1999.
[27] Wittmann R. and Zitterbart M., *Multicast Communication Protocols, Programming and Applications*, Morgan Kaufmann Publishers Inc., 2000.

Kanimozhi Sakthivel received ME degree in Computer Science and Engineering from Anna University of Technology, Coimbatore. Currently she is pursuing PhD in IST department from Anna University, Chennai..

Velumadhava Rao Rajasekaran received the BE degree in Computer Science and Engineering from C.I.E.T, Coimbatore and ME degree in Software Engineering from Anna University, Chennai, India. He is currently working as an Assistant Professor in Rajalakshmi Institute of Technology, Chennai. He is also pursuing his Part Time Ph.D in Computer Science and Engineering from Anna University, Chennai.

Selvamani Kadirvelu received B.E degree in Electrical and Electronics Engineering from Annamalai University and M.E degree in Computer Science and Engineering from Bharathiyar University and Ph.D degree in Information and Communication Engineering from Anna University. He is currently working as Assistant Professor in College of Engineering, Guindy, Anna University, Chennai.

Kannan Arputharaj received the M.E degree Computer Science and Engineering from AnnaUniversity and Ph.D degree in Information and Communication Engineering from Anna University. He is currently working as Professor in IST, College of Engineering, Guindy, Anna University, Chennai.