Security Improvement in Group Key Management

Manisha Manjul*, Rakesh Kumar, and Rajesh Mishra

School of ICT, Gautam Buddha University, Greater Noida, India
{manisham,rmishra}@gbu.ac.in

Abstract. Multicast is a one to the group communication which have various challenges such as group key management, multicast receiver access control, multicast finger printing and multicast source authentication. Various protocols introduced by many researchers to minimize the lacks such as computational, communication, message size and storage overheads for group key management, but these proposed methods still have some lack as discussed above, while rekeying cost is also not less. Therefore to provide a solution of existing problem after leaving a group, there is a need for efficient and improved mechanism for group key management.

Keywords: group key management, multicast security, rekeying, communication overheads.

1 Introduction

Computer network is basically the combination of computers and different type of devices that are interfaced by various resources via communication channels that provide the communications among users and allows them facility of sharing the resources [23]. Multicast is one of the service that provide different type of ways for communication such as one to many, many to one, many to many.

Multicast refers to the transmission of a message from one sender to multiple receivers or from multiple Senders to multiple receivers [14]. The advantage of multicast is that, it enables the desired applications to service many users without overloading a network and resources in the server. If the same message is to be sent to different destinations, multicast is preferred to multiple unicast. Group Communication introduces the challenging issues relating to group confidentiality and key management, when a source that sends data to a set of receivers in a multicast session. The security of the session is managed by two main functional, first is Group Controller (GC) responsible for authentication, authorization and access control, where as second is known a Key Server (KS) responsible for the maintenance and distribution of the required key material [11].

Security is the one of the issue in the multicast. There are four type of multicast security such as multicast receiver access control, multicast source authentication, multicast fingerprinting and group key management. All these multicast security have

* Corresponding author.

K. Singh, A.K. Awasthi, and R. Mishra (Eds.): QSHINE 2013, LNICST 115, pp. 879–893, 2013.

some issues and researchers provided solution for multicast security issues [8]. In this paper the authors have focused on group key management on security related issues. As we discuss the type of multicast communication, there are three types of multicast communication *i.e.* one-to-many, many-to-many and many-to-one. In each type of communication the idea is the same; the sender directs all datagram to a single IP address once and the datagram are delivered to every member of the multicast group [22]. One-to-many communication there exists only one sender and more than one hosts that are listening to the senders datagram. This is natural way for all types of on demand and file distribution services. One of the examples of one-to-many communications is telecast movies and all kinds of TV material.

Many-to-many multicast communication occurs usually when we are dealing with group communication. Video conferencing and other conferencing services are the example of many to many communications. More specifically these might include online gaming and online mentoring systems.

Many-to-one type of communication is very useful when we are providing high availability resource discovery and data collection services to large amount users. Auctioning services is the example of many to one communication. Multicasting is achieved with special routers, which keep track of all the networks within its routing domain that contain multicast/host group members. The routers do not have to keep track of all the members of multicast group. They just need to know the networks towards which they should copy the multicast datagram. In principle, the sender doesn't have to keep track of all the recipients either. But when we keep in mind the nature of most multicasting services, in practice there has to be some entity on behalf of service provider that registers all the receiving parties.

Multicast communication suffers from receiver access problem due to forward secrecy, backward secrecy. The group key management is an efficient mechanism to handle this situation. But there are many factors which effect the communication [6, 12, 13, and 14], computation overhead, message size, storage overhead, these factors are as following:

a. Heterogeneous nature of the group membership affects the possible type of encryption algorithm to be used, and the length of the key that can be supported by an end user.
b. The cost of setting up and initializing the entire system parameters, such as selection of the group controller (GC), group announcement, member join and initial key distribution.
c. Administrative policies, such as those defining which members have the authorization to generate keys.
d. Required level of performance of parameters, such as session sustainability, and key generation rates.
e. Required additional external support mechanisms, such as the availability of a certificate authority (CA).

Therefore it is required efficient group key management approach to secure the system and reduce the overhead in the existing approach [9]. Existing key graph [17]

proposes the extension of the binary key tree to 4-ary key tree and 4-ary key tree overcome the problem of re-keying in terms of height of the key tree. Using a greater degree reduces the height of the key tree and, as a result, improves re-keying performance. Performance of re-keying measured in terms of computation overhead, communication overhead, message size and storage overhead. Really, optimal results are gained when the tree has a degree of 4. In the figure 1(a) illustrates the logical key tree with two nodes when there are seven joining members (u_1 through u_7). When u_8 joins, the key server first attaches it to node $K_{1,2}$ as shown in figure 1(b), and then, changes the group key K_G and the node key $K_{1,2}$ to $K`_G$ and $K`_{1,2}$ respectively. For delivering them, each new key is encrypted with the previous one (K_G and $K_{1,2}$ respectively), and a set of them are sent by multicast for existing members. For the new member they are sent by unicast being encrypted with its session key. On the other hand, when a member leaves the group, new keys are encrypted by their corresponding child keys, and a set of them are sent for remaining members by multicast. For example, when member u_8 leaves the group shown in figure 5(b), the key server changes $K`_{1,2}$ and $K`_G$ to $K``_{1,2}$ and $K``_G$ respectively. Then, it delivers $K``_{1,2}$ for $\{u_5,u_6,u_7\}$ being encrypted by $K_5$, $K_6$ and $K_7$, and $K``_G$ for $\{u_1,u_2,u_3,u_4\}$ and $\{u_5,u_6,u_7\}$ being encrypted by $K_{1,1}$ and $K``_{1,2}$ respectively. A set of these keys are sent by one multicast message.

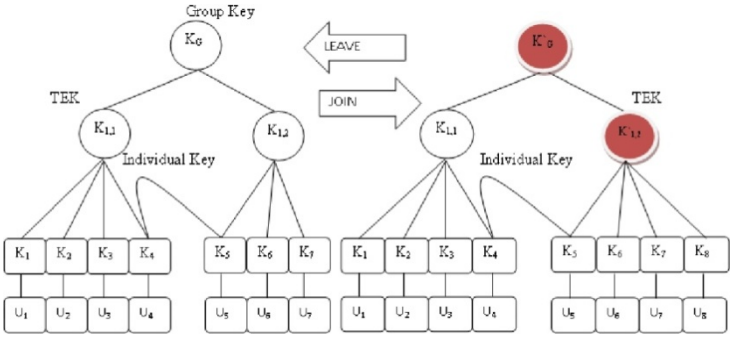

Fig. 1. Logical Key Tree for Key Graph (a) 4-Ary Tree for Seven Members (b) 4-Ary Tree for Eight Members

2 Proposed Protocol

It is based on the idea of key graph that manages the whole group on the basis of logical 4-ary key tree or key tree is the extended version of binary tree. In this protocol, the authors have divided whole group in several subgroups and every subgroup organized in a logical key hierarchy as in 4-ary key tree which reduce the complexity for a member join or leave from O (m) to O (log4n/m). The members in each subgroup contribute with each other to generate the subgroup key. This process delegates the key update process at a leave Process from the key server side to the member side. The proposed protocol works in a hierarchy of two levels of controllers;

the first for the group controller (GC) [3, 7] and the second is the subgroup controller (SC). The GC shares a symmetric key with all SCs which are trusted entities. The role of the SCs is to translate the data coming to their subgroups. Each SC works as the server of its subgroup. Figure 2: Illustrate the structure of proposed protocol.

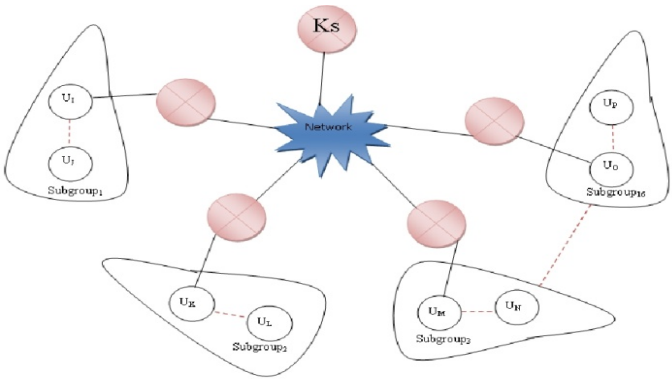

Fig. 2. Network Structure of Our Approach

The main objective of this protocol is to management a symmetric key between all group members in order to preserve the security of group communication [10]. In case of dynamicity 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 structure of the subgroup hierarchy in the proposed protocol is shown in figure 6. The subgroup is organized in a hierarchy like the LKH approach [18] and KS is the key of the group key. For the Process of the proposed protocol are following:

i. In this approach key server is a trusted entity which responsible for generate required keys and for distributing those keys to valid group members as shown in the figure 6 and Every member of the group has IGMP membership [1, 2, 16], when a new member joins a group; it sends an IGMP membership report message to its neighboring router to have the multicast data delivered from a multicast sender. Other side, the member sends a join request message to the key server to obtain the group key by which the multicast data is encrypted. This is different from other LKH approaches, in term to handle a large number of members efficiently; our approach divides group members into subgroups. For example 256 members are divided into 16 subgroups as shown in figure 3.
ii. Our approach applies the concept of key tree in LKH to the subgroups. In the logical key tree, leaf nodes correspond to subgroups, not individual members. Similar to other LKH approaches, the root node corresponds to the group key, and the intermediate nodes correspond to traffic encryption key (TEK) used for key transfers.
iii. The division of group members into subgroups is performed so that a balanced tree is constructed. In this case, by dividing n (256) members into subgroups whose size is m (16) members, we will have $\Gamma n/m \urcorner$ subgroups, and the height

of the tree will be $\log_4 \lceil n/m \rceil$. For example the group divided into 16 subgroups from 1 to 16 subgroups and height of the tree is 3 as shown in figure 3.

iv. When a member joins a group, it is allocated to a subgroup. At this time, the member obtains the following three kinds of key information from the key server.

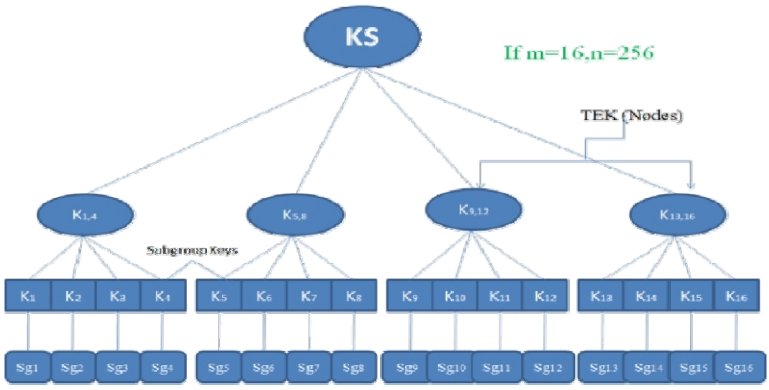

Fig. 3. Logical Key Tree in Proposed Approach

2.1 Design Methodology

Following steps have been involved in the designing stage.

2.1.1 Key Generation

As mentioned above, we use the modular exponential function as a one-way function. Since p is a large prime and g is the primitive element of multiplicative group $\boldsymbol{Z_p^*}$ it is computationally difficult to determine α given g and gα (mod p). Based on this property, the subgroup keys, the node keys and the group key are organized as follows.

First of all, the member secret α_j^i is selected under the condition that $2 \leq \alpha_j^i \leq p - 2$ and gcd $(\alpha_j^s, p-1)=1$.

The server secret the server secret α_j^s is selected under the condition that is selected under the condition that $2 \leq \alpha_j^s \leq p - 2$.

Using those secrets, the subgroup key for subgroup j is calculated by $Kj \equiv g\alpha_j^1 \alpha_j^2 \ldots\ldots\ldots\ldots \alpha_j^m \alpha_j^s (\text{mod } p)$. The node keys and the group key are organized by multiplying the exponents [20, 24] of its two child node keys (or the subgroup keys) in the logical key tree.

In order to illustrate the algorithm for re-keying, we use a simple example of multicast group divided into 16 subgroups; subgroup 1 to 16 with m members and subgroup 16 with m -1 members respectively. Figure.3 depicts the logical key tree for this group. The members of subgroups 1,2,3,4 own subgroup keys K1, K2, K3 and K4

respectively, node key K1, 4. The members of subgroups 5,6,7,8 own subgroup keys K5, K6, K7 and K8 respectively, node key K5, 8. The members of subgroups 9,10,11,12 own subgroup keys K9, K10, K11 and K12 respectively, node key K9, 12. The members of subgroups 13,14,15,16 own subgroup keys K13, K14, K15 and K16 respectively, node key K13,16 and group key KG. In this process key server used pre-computational function (PK) for calculating key when member join or leave the group and by using this pre-computational function process, we have minimized the computational cost during key generation and the keys are calculated as follows:

$K1 \equiv g \propto_1^1 \ldots\ldots \propto_1^{m-1} \propto_1^m \propto_1^{!s} \pmod p$

$K2 \equiv g \propto_2^1 \ldots\ldots \propto_2^{m-1} \propto_2^m \propto_2^{!s} \pmod p$

$K3 \equiv g \propto_3^1 \ldots\ldots \propto_3^{m-1} \propto_3^m \propto_3^{!s} \pmod p$

$K4 \equiv g \propto_4^1 \ldots\ldots \propto_4^{m-1} \propto_4^m \propto_4^{!s} \pmod p$

$K5 \equiv g \propto_5^1 \ldots\ldots \propto_5^{m-1} \propto_5^m \propto_5^{!s} \pmod p$

$K6 \equiv g \propto_6^1 \ldots\ldots \propto_6^{m-1} \propto_6^m \propto_6^{!s} \pmod p$

$K7 \equiv g \propto_7^1 \ldots\ldots \propto_7^{m-1} \propto_7^m \propto_7^{!s} \pmod p$

$K8 \equiv g \propto_8^1 \ldots\ldots \propto_8^{m-1} \propto_8^m \propto_8^{!s} \pmod p$

$K9 \equiv g \propto_9^1 \ldots\ldots \propto_9^{m-1} \propto_9^m \propto_9^{!s} \pmod p$

$K10 \equiv g \propto_{10}^1 \ldots\ldots \propto_{10}^{m-1} \propto_{10}^m \propto_{10}^{!s} \pmod p$

$K11 \equiv g \propto_{11}^1 \ldots\ldots \propto_{11}^{m-1} \propto_{11}^m \propto_{11}^{!s} \pmod p$

$K12 \equiv g \propto_{12}^1 \ldots\ldots \propto_{12}^{m-1} \propto_{12}^m \propto_{12}^{!s} \pmod p$

$K13 \equiv g \propto_{13}^1 \ldots\ldots \propto_{13}^{m-1} \propto_{13}^m \propto_{13}^{!s} \pmod p$

$K14 \equiv g \propto_{14}^1 \ldots\ldots \propto_{14}^{m-1} \propto_{14}^m \propto_{14}^{!s} \pmod p$

$K15 \equiv g \propto_{15}^1 \ldots\ldots \propto_{15}^{m-1} \propto_{15}^m \propto_{15}^{!s} \pmod p$

$K16 \equiv g \propto_{16}^1 \ldots\ldots \propto_{16}^{m-1} \propto_{16}^{!s} \pmod p$

$K1,4 \equiv g(\pi_i\alpha_1)(\pi_i\alpha_2)(\pi_i\alpha_3)(\pi_i\alpha_4) \pmod p$

$K5,8 \equiv g(\pi_i\alpha_5)(\pi_i\alpha_6)(\pi_i\alpha_7)(\pi_i\alpha_8) \pmod p$

$K9,12 \equiv g(\pi_i\alpha_9)(\pi_i\alpha_{10})(\pi_i\alpha_{11})(\pi_i\alpha_{12}) \pmod p$

$KG \equiv (PK_{13,16}PK_{9,12}PK_{5,8})^{\alpha_{16}^1 \ldots\ldots\ldots\ldots \alpha_{16}^{m-1}} \alpha_{16}^{!s} \pmod p$

2.1.2 Join Process

We now use to explain how re-keying is done when a new member joins the multicast group. In this process key server used pre-computational function for calculating key when member join or leave the group and this pre-computational function process minimized the computational cost during key generation. The procedure is as follows:

i. When key server receives a join request, it authenticates the member [5]. This may be done by the conventional approach such as remote authentication dial in user service (RADIUS) [4, 21], and we do not discuss this procedure. If required, the key server assigns the session key, and sends it to the member.
ii. The key server determines the subgroup for the new member and assigns the identity within the subgroup. In this example, the new member belongs to subgroup 16 and its identity is m. At this time, the path set for subgroup16, the keys K13, K14, K15, K16 and KG need to be changed to new ones.

iii. The key server assigns member secret α_{16}^{m} to u_{16}^{m}, and calculates its inverse value α_{16}^{-m} as well.

iv. The key server changes the server secret assigned to subgroup 16 from α_{16}^{s} to $\alpha_{16}^{!s}$.

v. The key server updates K16, K13,16 and KG to K`16, K`16 and K`G using α_{16}^{m} and $\alpha_{16}^{!s}$ in the following way.

$$K`16 \equiv g \propto_{16}^{1} \ldots\ldots \propto_{16}^{m-1} \propto_{16}^{m} \propto_{16}^{!s} \pmod p$$

$$K`13,16 \equiv g(\pi_i\alpha_{13})(\pi_i\alpha_{14})(\pi_i\alpha_{15})(\pi_i\alpha_{16}) \pmod p$$

$$K`G \equiv (PK_{13,16}PK_{9,12}PK_{5,8})^{\alpha_{16}^{1}\ldots\ldots\ldots\ldots\alpha_{16}^{m-1}} \propto_{16}^{m} \alpha_{16}^{!s} \pmod p$$

vi. The key server encrypts {K`16, K`13,16, K`G},and the inverse values of the other members in that subgroup, $\alpha_{16}^{-1} \ldots\ldots\ldots \alpha_{16}^{-m-1}$ by Kp_{16}^{m} than it sends this encrypted message through unicast to $\propto_{16}^{m}$. It has been given below:

$S \xrightarrow{\text{Unicat}} \{\propto_{16}^{m}\}:\{(\text{K`16, K`13,16, K`G, } \alpha_{16}^{-1} \ldots\ldots\ldots \alpha_{16}^{-m-1})Kp_{16}^{m}\}.$

vii. Server encrypts $\propto_{16}^{-m}$, and K`16 by K16for subgroup 16, K`13,16 by K13,16 for subgroup 13,14,15, K`G by KG for subgroup 1 to 12,and distributes these encrypted keys through multicast for existing members. This process describe as following:

$S \xrightarrow{\text{Multicast}} \{\text{Existing Members}\}$

$\{(\alpha_{16}^{-m}, \text{K`16) K16,( K`13,16) K13,16 ,(K`G) KG}\}.$

In this process, each updated key is encrypted by the previous one for existing members, and as a result, only the members who know the corresponding previous keys can decrypt the encrypted message containing the new keys.

2.1.3 Leave Process

When user u_{16}^{m} leaves the group then all member of the group affected by this change and key server changes the group key or path key such as K`16 to K``16, K13,16 to K``13,16 and KG to K``G. According to our protocol, these updated keys do not need to be sent to the remaining members. Instead, the key server just prepares one message for subgroup 16 indicating u_{16}^{m} leaves and delivers α_{16}^{-m} for subgroup 1 to 15.

The value of α_{16}^{-m} is encrypted into multiple copies by K15 and K13,16, for subgroup 15 and 1 to 14 respectively. The key server sends this message through multicast. This process describe as following:

$S \xrightarrow{Multicast} \{\text{Remaining members}\}$

$:\{(\alpha_{16}^{-m}) \text{ K15, } (\alpha_{16}^{-m}) \text{ K13,16}\}.$

When the remaining members receive this message, they decrypt it by the corresponding keys and then Use u_{16}^{m} to update those keys.

$K``16 = (K`16)\ \alpha_{16}^{-m} \pmod p$

$\equiv g\ \alpha_{16}^{1} \ldots\ldots \alpha_{16}^{m} \alpha_{16}^{-m} \alpha_{16}^{ls} \pmod p$

$K``_{16} \equiv g\ \alpha_{16}^{1} \ldots\ldots \alpha_{16}^{m-1} \alpha_{16}^{ls} \pmod p$

$K``_{13,16} \equiv (K`_{16})\ \alpha_{16}^{-m} \pmod p$

$\equiv g(\pi_i \alpha_{13})(\pi_i \alpha_{14})(\pi_i \alpha_{15})(\pi_i \alpha_{16}) \pmod p$

$= g(\pi_i \alpha_{13})(\pi_i \alpha_{14})(\pi_i \alpha_{15})(\pi_{i-1} \alpha_{16}) \pmod p$

$K``_G \equiv (K`_G)\ \alpha_{16}^{-m} \pmod p$

$\equiv (PK_{13,16} PK_{9,12} PK_{5,8})^{\alpha_{16}^{i}} \ldots\ldots\ldots\ldots \alpha_{16}^{-m} \alpha_{16}^{m} \alpha_{16}^{ls} \pmod p$

$\equiv (PK_{13,16} PK_{9,12} PK_{5,8})^{\alpha_{16}^{i}} \ldots\ldots\ldots\ldots \alpha_{16}^{-m} \alpha_{16}^{ls} \pmod p$

As we notice, the key server does not need to generate new keys (TEK and group key) after a leave. Instead, it just sends the inverse value of leaving member to remaining members. Then, the remaining members update the necessary keys. In this way, updating the keys after a leave is shifted to member's side which improves the efficiency of re-keying at leave.

3 Comparison with Exist One

In the comparison, n denotes the number of members in the group, the number after a join and before a leave in a strict numbers, called group size. We define m as the number of members in a subgroup, called subgroup size only for our proposal. We also show some numerical results for the overhead by changing the group sizes from 16 to 1048576. To evaluate our proposal, we use subgroup size is i.e. 256.

We are using a binary tree for LKH and OFT, and our proposal is based on 4-ary key tree. In case of simple app. height of the tree h=2 and height LKH=OFT=$\log_2 n$. The height of the key tree for our proposal under the condition of $n \leq m$ will be equal to 1 and under the condition of $n>2m$ will be $\log 4\ \lceil n/m \rceil$. In general, the number of node keys is proportional to the height of the key tree in LKH based protocols, in other words, a protocol with smaller height has fewer nodes along the path. In our approach we have minimized the height of the tree along the key path and number of key generation, encryption/decryption also minimized. Due to this reason the performance of the system will be improve.

3.1.1 Computational Overhead

Computational overhead depending on the Key generation overhead and encryption/decryption overhead as following:

3.1.1.1. Key Generation Overhead. It is the overhead at the key server and member node along the path to the root at each join or leave. The number of key generations at the key server is almost equal to the height of the key tree. First of all, Simple App. has the smallest overhead at the key server both join and leave process. Our approach minimizes number of key generation at the key server both join and leave process as compare to LKH and OFT. By contrast, because of smaller size of h_{sg}, the key server generates fewer keys at join. Most importantly, the key server does not need to generate new keys for the members at leave.

On the other hand in simple application and LKH, a member node does not generate any keys by it at each event, but in OFT the new member at join and a remaining member node along the path at leave need generate new node keys by mixing two hash values. At a member leave process the group and subgroup controller doesn't generate any keys. Instead it multicasts the identity of the leaving member to all the group and subgroup members to be factored from the subgroup key by using the leaving member's inverse value. Figure 4(a) and 4(b) shows comparative result of number of key generation overhead on the basis of group size and number of key generated at the key server.

From the figure 4(a) one can notice that the proposed protocol has minimized overhead at the join process because the key server reduced the height of key tree by using 4-ary key tree. From the figure 4(b) one can notice that the proposed protocol has the smallest overhead at the leave process because the key server doesn't generate any keys in that case.

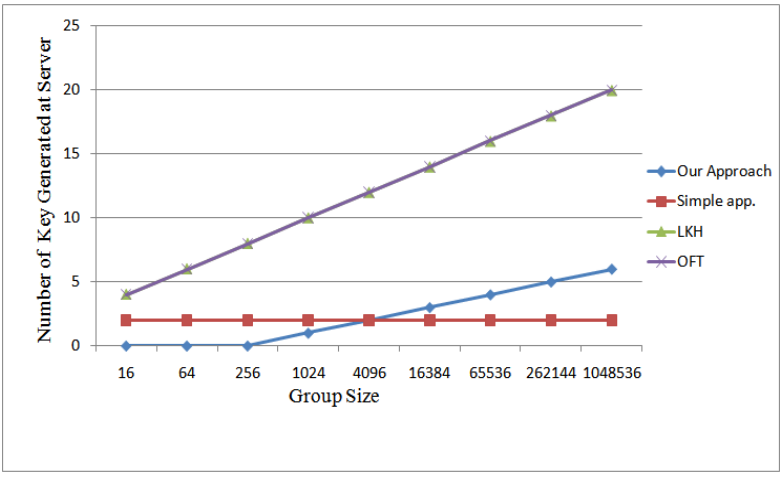

Fig. 4. (a): Key Generation Overhead at the Key Server during Join Process

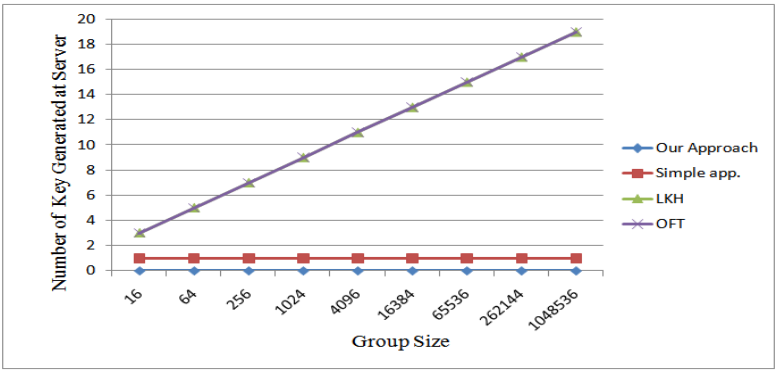

Fig. 4. (b): Key Generation Overhead at the Key Server during Leave Process

Figure 5(a) and 5(b) shows comparative result of number of key generation overhead on the basis of group size and number of key generated at the member node.

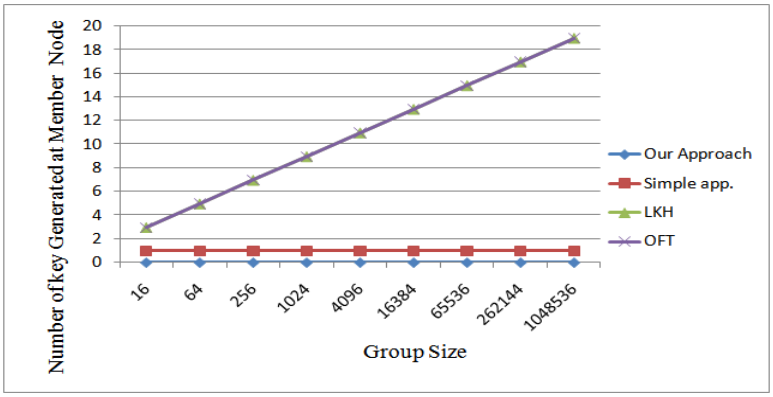

Fig. 5. (a): Key Generation Overhead at the Member Node during Join Process

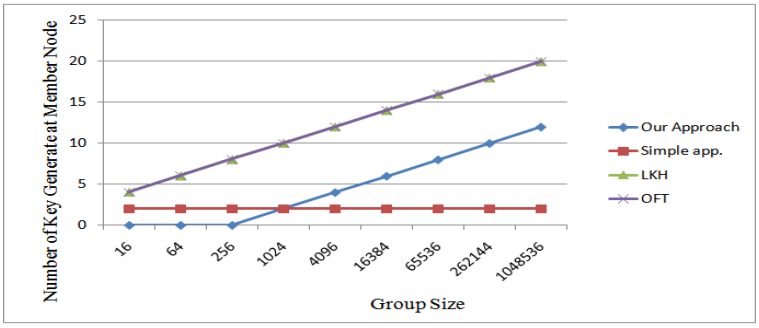

Fig. 5. (b): Key Generation Overhead at the Member Node during Leave Process

The key generation overhead for our protocol is 0 at join, but proportional to the height of the key tree at leave as shown in the table 1 at member node. In fact, a member node renews the node keys along the path to the root by modular exponentiation.

3.1.1.2. Encryption/Decryption Overhead. Encryption overhead at the key server (left side) and decryption overhead at a member node (right side) at each join and leave. The values for join process at the key server are the sums of the number of encryptions for existing members and a new member. Although the overhead at the key server for Simple App. is the smallest value at join, it is the largest one of all at leave. In simple App. the key server has to perform n-1encryptions for the remaining members at leave, when n are the number of members. This is the problem we mentioned in Section 4; in which other protocols have tried to solve this problem by introducing the 4-ary key tree.

At the key server, LKH involves two separate encryptions per node, one for each of its two children, compared to OFT which involves one encryption per node. Therefore, even with the same height of LKH = height of OFT, the encryption overhead for LKH is larger than of that for OFT. By contrast, because of the small height of subgroup (hsg) and small height of group hg, the key server performs fewer encryptions at join and leave for our protocol compared with LKH, OFT. The encryption /decryption formula for proposed approach and previous approaches as shown in the table 3, according to our approach number of encryption will be minimize at the key sever at the time of both join and leave process. On the other hand number of decryption at the member node also minimized at the joining time as compared to LKH and OFT.

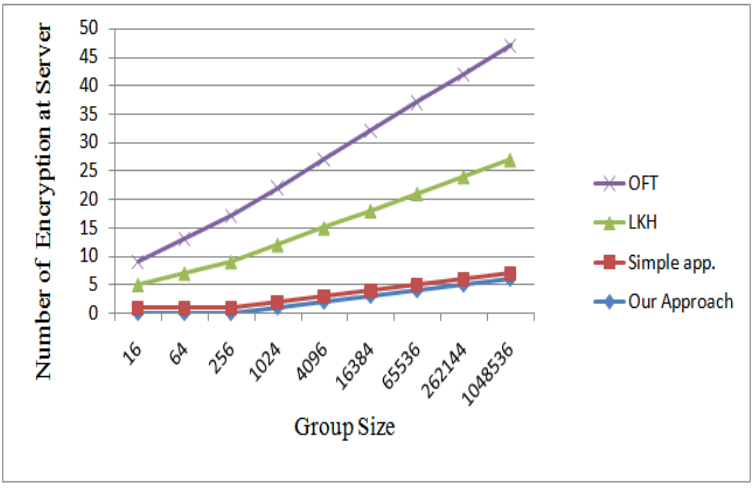

Fig. 6. (a): Number of Encryption at the Key Server during Join Process

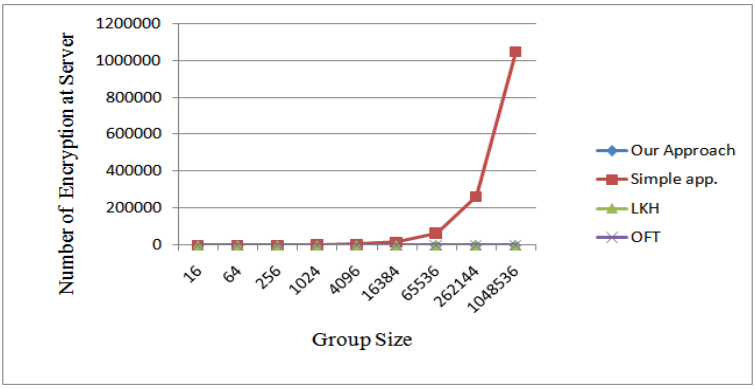

Fig. 6. (b): Number of Encryption at the Key Server during Leave Process

Figure 6 (a) and 6(b) show the number of encryption at the key server at join and leave process respectively. At a member node, the decryption overhead at join is proportional to height of the key tree for all protocols, in which simple app. has smallest overhead, but LKH and OFT have the largest overhead. Because of height of LKH and OFT is largest as compare to propose approach. So, that proposed approach minimize the number of decryption at the member node.

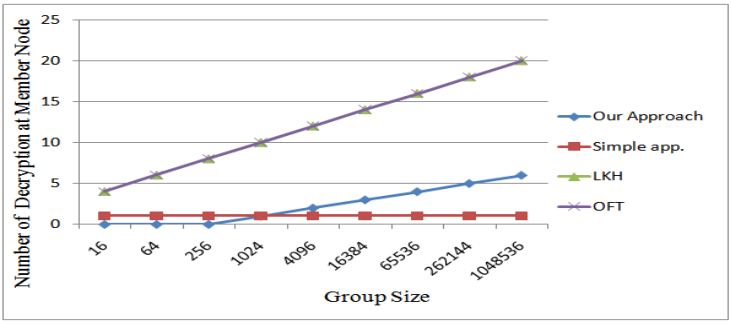

Fig. 7. (a): Number of Decryption at the Member Node during Join Process

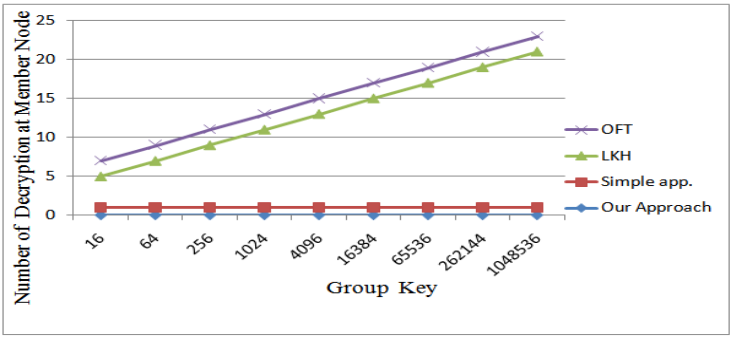

Fig. 7. (b): Number of Decryption at the Member Node during Leave Process

Figure 7 (a) and 7(b) show the number of encryption at the member node at join and leave process respectively.

3.1.1.3. Communication Overhead. Communication overhead at join and leave for multicast communication as shown in table 4. As described above, this is measured by the number of transmitted control messages. Figure 8(a) and 8(b) illustrate numerical results at join and leave, respectively. At join, Simple App., and our proposal have a small overhead. LKH has the largest overhead and OFT has half of that. On the other hand, at leave, Simple App. has an extremely large overhead and our protocol have a small overhead.

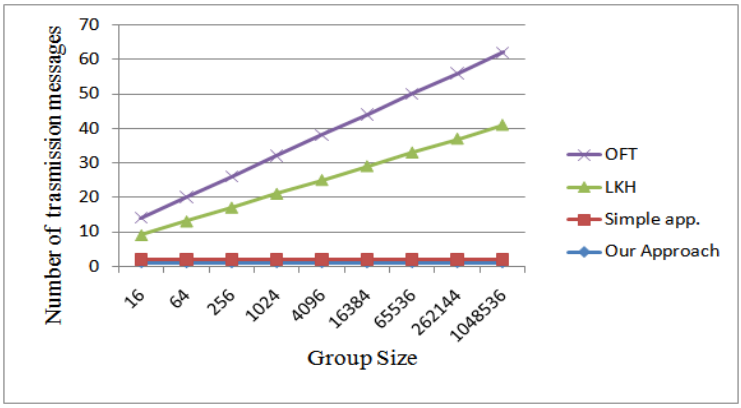

Fig. 8. (a): Communication Overhead at Server during Join Process

On the basis of comparative results our protocol the best in terms of the communication overhead. Because of this is send all key information in one multicast message to existing members at join, and to remaining members at leave. It should be noticed that the message size is different as shown later; it is bigger for LKH than for our protocol.

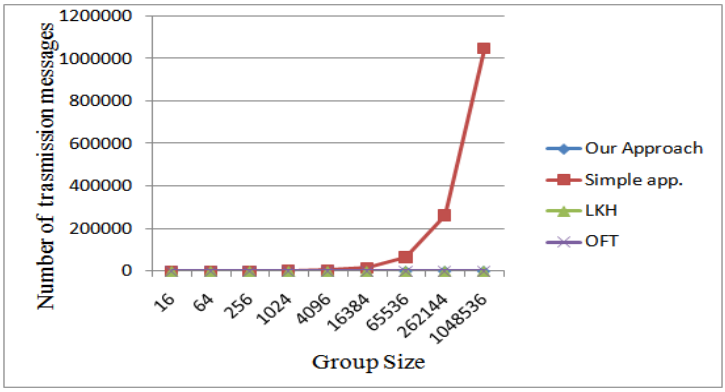

Fig. 8. (b): Communication Overhead at Server during Leave Process

4 Conclusion

We have discussed different type of security in the multicast such as multicast receiver access control, multicast source authentication, multicast fingerprinting and group key management. We have selected group key management area of multicast security and we can say group key management is the part of multicast security. We have found different type of issues such as computational overhead, communicational overhead, message size and storage overhead in the group key management and many researchers provided LKH, OFT etc. solutions for these issues but these issues are not solved. Therefore, in this dissertation, we have proposed a security improvement in group key management approach to solve the problem of distributing a symmetric key between the whole group members for secure group communication. The performance of the proposed protocol is compared with that of the Simple App., OFT and LKH protocols. The comparison is undertaken according to the computational overhead, communication overhead, storage overhead, and message size. The results show that the proposed protocol improves the group performance in terms of computation overhead, message size and communication overhead.

References

[1] Je, D.-H., Lee, J.-S., Park, Y., Seo, S.-W.: Computation-and-storage-efficient key tree management protocol for secure multicast communications. Computer Communications 33(2), 136–148 (2010)

[2] Pour, A.N., Kumekawa, K., Kato, T., Itoh, S.: A hierarchical group key management scheme for secure multicast increasing efficiency of key distribution in leave operation. Elsevier, Computer Networks 51(17), 4727–4743 (2007)

[3] Chen, X., Ma, B.N.W., Yang, C.: M-CLIQUES: Modified CLIQUES key agreement for secure multicast. Computers & Security 26(3), 238–245 (2007)

[4] Sun, Y(L.), Ray Liu, K.J.: Hierarchical Group Access Control for Secure Multicast Communications. IEEE/ACM Transactions on Networking 15(6) (December 2007)

[5] Lee, P.P.C., Lui, J.C.S., Yau, D.K.Y.: Distributed Collaborative Key Agreement and Authentication Protocols for Dynamic Peer Groups. IEEE/ACM Transactions on Networking 14(2) (April 2006)

[6] Abdellatif, R., Aslan, H.K., Elramly, S.H.: New Real Time Multicast Authentication Protocol. International Journal of Network Security 12(1), 13–20 (2011)

[7] Zheng, S., Manz, D., Alves-Foss, J.: A communication computation efficient group key algorithm for large and dynamic groups. Elsevier, Computer Networks 51(1), 69–93 (2007)

[8] Wallner, D., Harder, E., Agee, R.: Key Management for Multicast: Issues and Architecture. National Security Agency (June 1999), RFC 2627

[9] Saroit, I.A., El-Zoghdy, S.F., Matar, M.: A Scalable and Distributed Security Protocol for Multicast Communications. International Journal of Network Security 12(2), 61–74 (2011)

[10] Baugher, M., Canetti, R., Dondeti, L., Lindholm, F.: Multicast Security (MSEC) Group Key Management Architecture. RFC 4046 (April 2005)

[11] Challal, Y., Seba, H.: Group Key Management Protocols: A Novel Taxonomy. International Journal of Information Technology 2(1) (2005) Issn: 1305-2403
[12] Wong, C.K., Gouda, M., Lam, S.S.: Secure Group Communications Using Key Graphs. IEEE/ACM Transactions on Networking 8(1) (February 2000)
[13] Jabeenbegum, S., Purusothaman, T., Karthi, M., Balachandar, N., Arunkumar, N.: An Effective Key Computation Protocol for Secure Group Communication in Heterogeneous Networks. IJCSNS International Journal of Computer Science and Network Security 10(2) (February 2010)
[14] Srinivasan, R., Vaidehi, V., Rajaraman, R., Kanagaraj, S., Chidambaram Kalimuthu, R., Dharmaraj, R.: Secure Group Key Management Scheme for Multicast Networks. International Journal of Network Security 11(1), 33–38 (2010)
[15] Ng, W.H.D., Howarth, M., Sun, Z., Cruickshank, H.: Dynamic Balanced Key Tree Management for Secure Multicast Communications. IEEE Transactions on Computers 56, 590–605 (2007)
[16] Lu, H.: A Novel High-Order Tree for Secure Multicast Key Management. IEEE Transactions on Computers 54, 214–224 (2005)
[17] Wong, C.K., Gouda, M., Lam, S.S.: Secure group communications using key graphs. IEEE/ACM Transactions on Networking 8(1), 16–30 (2000)
[18] Wallner, D., Harder, E., Agee, R.: Key Management for Multicast: Issues and architectures. National Security Agency (June 1999), RFC 2627
[19] Dierks, T., Rescorla, E.: The Transport Layer Security (TLS). Protocol Version 1.1 (April 2006), RFC 2346
[20] Stinson, D.R.: Cryptography Theory and Practice", Second edition. Chapman and Hall/CRC Press, 155–175 (2002)
[21] Rigney, C., Willens, S., Rubens, A., Simpson, W.: Remote Authentication Dial in User Service (RADIUS) (June 2000), RFC 2865
[22] Deering, S.: Host Extensions for IP Multicasting. RFC 1112 (August 1989)
[23] Tanenbaum, A.: Computer Networks, 4th edn. Prentice Hall (2009)
[24] Stallings, W.: Cryptography and Network Security Principles and Practices, 4th edn., p. 592 (November 16, 2005)