Society for Science and Education United Kingdom
TNC Transactions on Networks and Communications
Volume 2, Issue 1
ISSN: 2054 - 7420

Secure Group Communication based on Elliptic Curve Cryptography

Manisha Manjul , Rajesh Mishra
Department of CSED, School of ICT, Gautam Buddha University
Greater Noida, India

ABSTRACT

Group key management is an important functional building block for any secure multicast architecture. In this regards, it is identified some security issues in the group key management when a user join or leave the group then forward secrecy and backward secrecy issues comes in the multicast networks. This paper provides an efficient and improve mechanism for group key management solutions with computational and communication overhead are less while rekeying cost also minimize. The proposed approached is known as Elliptic Curve Cryptography (ECC) based secure Group Communication.

Keywords: Group key management, Multicast security, Key management protocol, Elliptic Curve Cryptography.

1 INTRODUCTION

Computer network is one of the parts of modern era. A computer network is basically the combination of computers and different type of devices that are interfaced by various resources i.e. communications channels that provide the communications among users and allows them facility of sharing the resources [63]. 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 [1, 2]. 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, whereas

DOI: 10.14738/tnc.21.7
Publication Date:14th February 2014
URL: http://dx.doi.org/10.14738/tnc.21.7

second is known a Key Server (KS) responsible for the maintenance and distribution of the required key material [7]. Different types of multicast security are given below:

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 some issues and researchers provided solution for multicast security issues. We have selected group key management research area and rests of the multicast group security discuss in with details the next section.

1.1 Group Key Management

Multicasting optimizes the usage of network links by sending one packet to the whole multicast group instead of sending a separate packet to each receiver; it is very natural to encrypt the multicast packets only once too. This leads us to the introduction of the concept of the group key [26]. A group key is a common security key that is distributed to the whole secure group. Group key can be used to encrypt a multicast packet and the valid multicast users are able to decrypt the packet using the group key. The usage of group key itself is very simple, but the distribution and revocation situations are problematic [47]. When a member of a multicast group leaves the group all the common group keys have to be renewed so that the leaving member cannot receive any later messages destined to the group. As we do not have any more secure group keys which we could use to distribute the new group keys to each member we have to use member specific keys or some other way to create a new group key. From performance standpoint the key renewal operation becomes very heavy if the number of group members is very large and only one server is responsible for doing it. To distribute new keys to every member separately is very ineffective In general we are able to identify characteristics for key distribution system that should be considered when we are selecting or defining a key management scheme.

In other words we can say that group key encryption used in which the multicast traffic is encrypted with a symmetric key and every authorized member of the group is given the decryption key. This becomes complicated when group membership is Dynamic. Upon a change in membership, it is often necessary to change the group key so that the leaving member cannot access new broadcasts or a new member cannot access old broadcasts. The term leave is used to describe the act of a voluntary or forced leave.

The phenomenal growth of the Internet in the last few years and the increase of bandwidth in today’s networks have provided both inspiration and motivation for the development of new services, combining voice, video and text over Internet Protocol. Although unicast communications have been predominant so far, the demand for multicast communications is increasing both from the internet service providers and from content or media providers and distributors. Indeed, multicasting is increasingly used as an efficient communication mechanism for group-oriented applications on the Internet such as video conferencing, interactive group games, video on demand, TV over Internet, e-learning, software updates, database replication and broadcasting stock quotes. Key management is the base for providing common security services (data secrecy, authentication and integrity) for group communication. The main goal of this research is to demonstrate how provably our proposed security improvement in group key management protocol can be combined with reliable group communication services to obtain provably efficient communication and computation costs [3].

In near future the networks will have bandwidths ranging from kilobits per second to gigabits per second. Networks will have different users and applications in heterogeneous network environments. There are two types of security issues in the group key management security requirement and quality of services. There are different types of security requirement in group key management such as forward secrecy**,** backward secrecy, collusion freedom, key independence and trust relationship [37]. Forward secrecy requires that users who left the group should not have access to any future key. This ensures that a member cannot decrypt data after it leaves the group. To assure forward secrecy, a re-key of the group with a new TEK after each leave from the group is the ultimate solution. Backward Secrecy requires that a new user that joins the session should not have access to any old key. This ensures that a member cannot decrypt data sent before it joins the group. To assure backward secrecy, a re-key of the group with a new TEK after each join to the group is the ultimate solution. Collusion Freedom: its requirements that any set of fraudulent users should not be able to deduce the current traffic encryption key. Key Independence: This ensures that any subset of a group keys must not be able to discover any other group key. Trust relationship: This is emphasized by definition of verifiable trust relationship that consists of two requirements: One as Group members are trusted not to reveal the group key or secret values that may lead to its computation to any other party, and another as group members must be able to verify the computation steps of the group key management protocol.

There are some important factors which affect the quality of group key management that are given such as Low Bandwidth Overhead and 1-Affects-N. Low Bandwidth Overhead**:** The re-key of the group should not induce a high number of messages, especially for dynamic groups. Ideally, this should be independent from the group size.

Affects-N*:* a protocol suffers from the 1-affects-n phenomenon if a single membership change in the group affects all the other group members. This happens typically when a single membership change requires that all group members commit to a new TEK

1.2 Background

First of all we are discussing type of multicast; there are three types of multicast communication: One-to-many, many-to-many and many-to-one. In every 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.

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 rooting 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. And this is essentially the case from the security point of view too

1. Group Management
2. Multicasting provides very dynamic framework for managing group members. Members can join and leave groups at any time with the help of Internet group management protocol (IGMP) [57]. A group can contain any number of member hosts. Also, there is no restriction on the number of groups that a host can belong to. IGMP operates only at LAN level between group hosts and closest LAN routers. In IPv4 IGMP is an extension to Internet Control Message Protocol (ICMP)
3. IGMP is an asymmetric protocol meaning that member hosts are the only ones sending IGMP messages. They communicate with the closest multicast router with join or leave group messages. The task of the router is then to forward the host's will upstream towards the multicast source or rendezvous point using multicast routing protocols. IGMP in its simplicity doesn't provide any means for authenticating hosts or users when they are requesting or quitting the membership of a group. For services like our imaginative Tatasky.com the authentication of service users is extremely important. So to make multicast services viable for providers and secure for users we need additional services and protocols.

1.2.1 Multicast Key Management

Secure multicast communication requires that the messages between group members are authenticated, encrypted and integrity checked [48]. The group members want to be sure that their communication is not eavesdropped and that they are sure whom they are really communicating with. The requested quality service level has to be guaranteed even though security services are in use in the network. And when the composition of the group is changing the group members and service providers want to be absolutely sure that members leaving the group cannot read the messages [23, 29].

1.2.2 Group Key Management

Group key management is an important functional building block for any secure multicast architecture. A group key management (GKM) protocol supports protected communication between members of a secure group. A secure group is a collection of members, who may be senders, receivers, or both receivers and senders to other members of the group. A group key management protocol helps to ensure that only members of a secure group can gain access to group data and can authenticate group data [9, 25, and 33]. There are many approaches exist [24][30][31][32] to provide the solution of group key management problems but mostly use the RSA and Deffi-Hellman approach to create and share the key between source and destination but these algorithms also increase the computing and communication overheads of system. Our proposed approach "Elliptic Curve Cryptography (ECC) based secure Group Communication is one option to reduce the computing and communication at source as well as destination. Section 2 is discussing the proposed approach.

2 PROPOSED WORK

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, computation overhead, message size, storage overhead [36], these factors are as following:

1. 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.
2. 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.
3. Administrative policies, such as those defining which members have the authorization to generate keys.
4. Required level of performance of parameters, such as session sustainability, and key generation rates.
5. Required additional external support mechanisms, such as the availability of a certificate authority (CA).

There are we require efficient group key management approach to secure the system and reduce the overhead in the existing approach [10, 29]. Exist key graph [52] proposed the extension of the binary key tree to 4-ary key tree. 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.

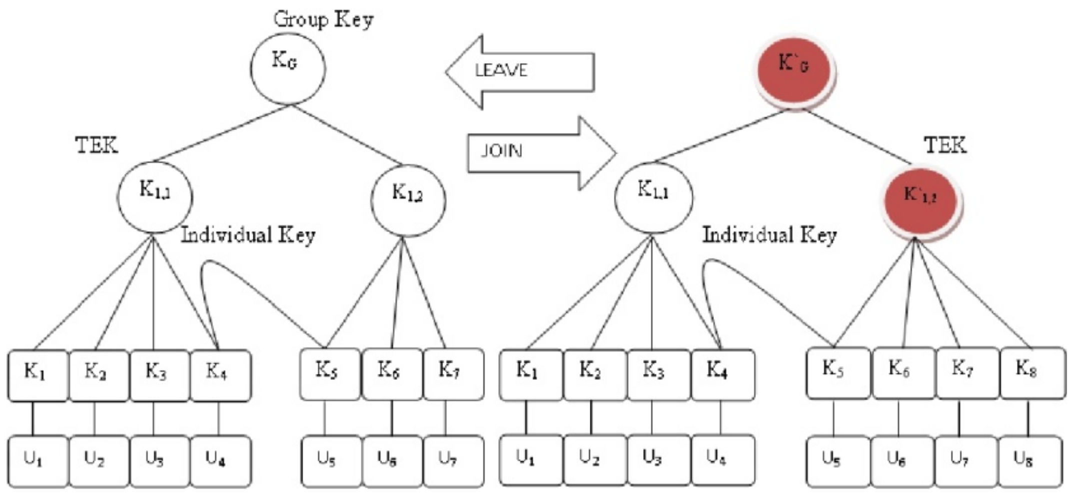

Figure 1: Logical Key Tree for Key Graph (a) 4-Ary Tree for Seven Members (b) 4-Ary Tree for Eight Members

2.1 Proposed System Model

The proposed protocol 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, we 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) 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.

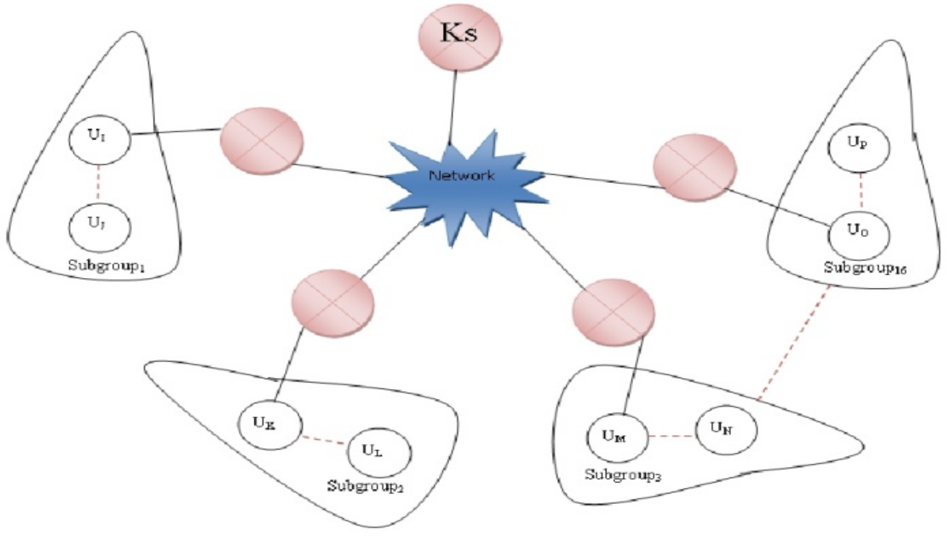

Figure 2: Group Formation Procedure

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. 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 2. The subgroup is organized in a hierarchy like the LKH approach [34, 53] and KS is the key of the group key. For the Process of the proposed protocol are following:

1. 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 2 and Every member of the group has IGMP membership, 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.
2. 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.
3. 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 $\lceil n/m \rceil$ 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.

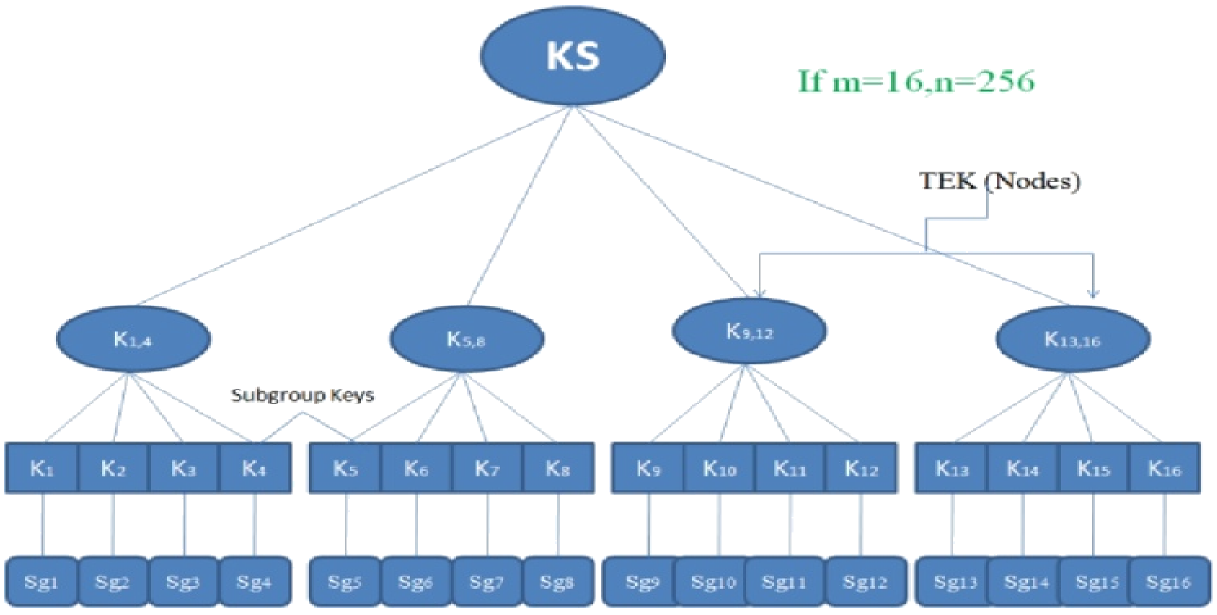
KS
If m=16,n=256
TEK (Nodes)
K1,4
K5,8
K9,12
K13,16
Subgroup Keys
K1 K2 K3 K4 K5 K6 K7 K8 K9 K10 K11 K12 K13 K14 K15 K16
Sg1 Sg2 Sg3 Sg4 Sg5 Sg6 Sg7 Sg8 Sg9 Sg10 Sg11 Sg12 Sg13 Sg14 Sg15 Sg16

Figure 3: Logical Key Tree in Proposed Approach

4. 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.

a. The private key: This key which is shared only with the key server is used to transfer information securely between the key server and the member. Moreover, this key may be a predetermined key, or assigned dynamically through a secure channel such as TLS [54].

b. The keys which are defined in accordance with the logical key tree: As described above, they include the subgroup key, the node keys and the group key, along the path from the leaf node (corresponding to the subgroup) to the root node. We call these keys the path set.

c. The key information of other members in the subgroup: This information is used to change the subgroup key, the node keys and the group key when a member in the subgroup leaves the multicast group. This information is called the inverse value in the rest of the paper. The details of these keys are described below in this section and in the next section. For generating subgroup keys, node keys and group keys, our approach uses the modular exponential calculation over the finite field [55] using member secrets and the server secrets. A member secret is a value which is assigned to each member at join and a server secret is a value which is assigned to each subgroup at join to ensure backward secrecy. These values are generated by the server at join (see next section). Moreover, each subgroup key is generated by using the member secrets and the server secret assigned to that subgroup with modular exponentiation. Next, the node keys and the group key are calculated from two child node keys in the logical key tree.

5. The group key and the subgroup key at join are updated as follow: First, the key server generates a member secret for the new member and finds its inverse value. Next, the corresponding subgroup key, where the new member is allocated, the TEK and the group key along the path are updated by adding the corresponding member secret and the new value of server secret to the exponent parts of those keys with modular exponential function. By changing the server secret at each join, the backward secrecy is satisfied.
6. In order to update the keys efficiently at leave, we use inverse values of member secrets. Any member in a subgroup is in-formed of the inverse values of all the other members in the sub-group except its own. By using the inverse values at leave, the path set in the subgroup whose member leaves can be updated by the remaining members themselves rather than delivered by the key server. New keys can be calculated by applying the in-verse value of the leaving member to the exponent part of the previous keys by modular exponentiation. For the members who do not belong to the same subgroup to which the leaving member belonged, the inverse value of the leaving member is delivered by the key server. This procedure satisfies the forward secrecy with low overhead.

2.2 Design Methodology

2.2.1 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 Z_p^* it is computationally difficult to determine α given g and g^{α} (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 (α_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 $K_j \equiv g\alpha_j^1\alpha_j^2 \ldots\ldots\ldots\ldots \alpha_j^m\alpha_j^s$(mod p). The node keys and the group key are organized by multiplying the exponents 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 K_1, K_2, K3 and K_4 respectively, node key $K_{1,4}$. The members of subgroups 5,6,7,8 own subgroup keys K_5, K_6, K_7 and K_8 respectively, node key $K_{5,8}$. The members of subgroups 9,10,11,12 own subgroup keys K_9, K_{10}, K_{11} and K_{12} respectively, node key $K_{9,12}$. The members of subgroups 13,14,15,16 own subgroup keys K_{13}, K_{14}, K_{15} and K_{16}

respectively, node key $K_{13,16}$ and group key K_G. 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:

$$K_1 \equiv g \propto_1^1 \ldots\ldots \propto_1^{m-1} \propto_1^m \propto_1^{!s} \quad (\text{mod } p)$$

$$K_2 \equiv g \propto_2^1 \ldots\ldots \propto_2^{m-1} \propto_2^m \propto_2^{!s} \quad (\text{mod } p)$$

$$K_3 \equiv g \propto_3^1 \ldots\ldots \propto_3^{m-1} \propto_3^m \propto_3^{!s} \quad (\text{mod } p)$$

$$K_4 \equiv g \propto_4^1 \ldots\ldots \propto_4^{m-1} \propto_4^m \propto_4^{!s} \quad (\text{mod } p)$$

$$K_5 \equiv g \propto_5^1 \ldots\ldots \propto_5^{m-1} \propto_5^m \propto_5^{!s} \quad (\text{mod } p)$$

$$K_6 \equiv g \propto_6^1 \ldots\ldots \propto_6^{m-1} \propto_6^m \propto_6^{!s} \quad (\text{mod } p)$$

$$K_7 \equiv g \propto_7^1 \ldots\ldots \propto_7^{m-1} \propto_7^m \propto_7^{!s} \quad (\text{mod } p)$$

$$K_8 \equiv g \propto_8^1 \ldots\ldots \propto_8^{m-1} \propto_8^m \propto_8^{!s} \quad (\text{mod } p)$$

$$K_9 \equiv g \propto_9^1 \ldots\ldots \propto_9^{m-1} \propto_9^m \propto_9^{!s} \quad (\text{mod } p)$$

$$K_{10} \equiv g \propto_{10}^1 \ldots\ldots \propto_{10}^{m-1} \propto_{10}^m \propto_{10}^{!s} \quad (\text{mod } p)$$

$$K_{11} \equiv g \propto_{11}^1 \ldots\ldots \propto_{11}^{m-1} \propto_{11}^m \propto_{11}^{!s} \quad (\text{mod } p)$$

$$K_{12} \equiv g \propto_{12}^1 \ldots\ldots \propto_{12}^{m-1} \propto_{12}^m \propto_{12}^{!s} \quad (\text{mod } p)$$

$$K_{13} \equiv g \propto_{13}^1 \ldots\ldots \propto_{13}^{m-1} \propto_{13}^m \propto_{13}^{!s} \quad (\text{mod } p)$$

$$K_{14} \equiv g \propto_{14}^1 \ldots\ldots \propto_{14}^{m-1} \propto_{14}^m \propto_{14}^{!s} \quad (\text{mod } p)$$

$$K_{15} \equiv g \propto_{15}^1 \ldots\ldots \propto_{15}^{m-1} \propto_{15}^m \propto_{15}^{!s} \quad (\text{mod } p)$$

$$K_{16} \equiv g \propto_{16}^1 \ldots\ldots \propto_{16}^{m-1} \propto_{16}^{!s} \quad (\text{mod } p)$$

$$K_{1,4} \equiv g(\pi_i\alpha_1)(\pi_i\alpha_2)(\pi_i\alpha_3)(\pi_i\alpha_4) \quad (\text{mod } p)$$

$$K_{5,8} \equiv g(\pi_i\alpha_5)(\pi_i\alpha_6)(\pi_i\alpha_7)(\pi_i\alpha_8) \quad (\text{mod } p)$$

$$K_{9,12} \equiv g(\pi_i\alpha_9)(\pi_i\alpha_{10})(\pi_i\alpha_{11})(\pi_i\alpha_{12}) \quad (\text{mod } p)$$

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

2.2.2 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:

1. When key server receives a join request, it authenticates the member. This may be done by the conventional approach such as *remote authentication dial in user service* (RADIUS) [56], and we do not discuss this procedure. If required, the key server assigns the session key, and sends it to the member.
2. 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.
3. The key server assigns member secret α_{16}^{m} to u_{16}^{m}, and calculates its inverse value α_{16}^{-m} as well.
4. The key server changes the server secret assigned to subgroup 16 from α_{16}^{s} to $\alpha_{16}^{!s}$.
5. 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.

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

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

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

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:

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

6. Server encrypts α_{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:

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

$$:\{(\alpha_{16}^{-m}, K`_{16}) K_{16},( K`_{13,16}) K_{13,16} ,(K`_{G}) K_{G} \}.$$

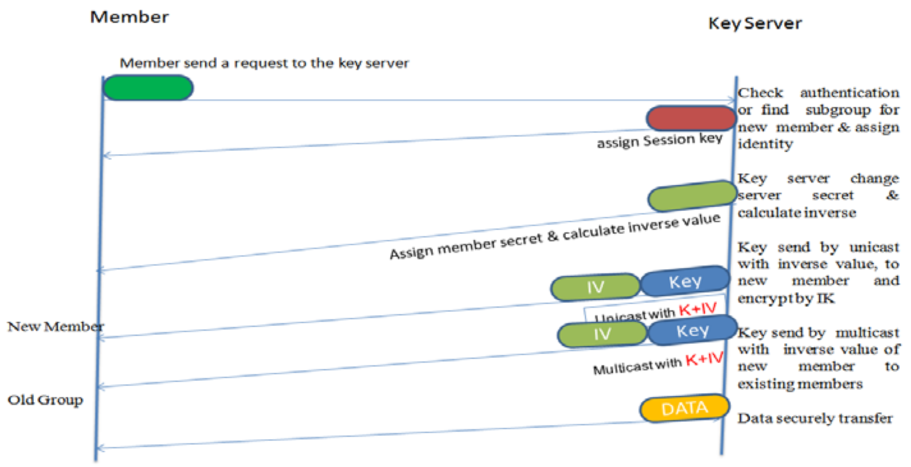
Member
Key Server
Member send a request to the key server
Check authentication or find subgroup for new member & assign identity
assign Session key
Key server change server secret & calculate inverse
Assign member secret & calculate inverse value
Key send by unicast with inverse value, to new member and encrypt by IK
IV
Key
Unicast with K+IV
New Member
IV
Key
Key send by multicast with inverse value of new member to existing members
Multicast with K+IV
Old Group
DATA
Data securely transfer

Figure 4: Joining Process of Proposed Approach

The complete process of join a group and assign the key is give in figure 4. 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.

2.2.3 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 $K``_{16,}$ $K_{13,16}$ to $K``_{13,16}$ and K_G 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 K_{15} and $K_{13,16,}$ for subgroup 15 and 1 to 14 respectively. The key server sends this message through multicast. This process describe as following:

$$S\xrightarrow{Multicast}\{\text{Remaining members}\}\ \{(\alpha_{16}^{-m})\ K_{15,}\ (\ \alpha_{16}^{-m})\ K_{13,16}\}.$$

And the complete process of join a group and assign the key is give in figure 5.

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

$$K``_{16} \equiv (K`_{16})\ \alpha_{16}^{-m} \qquad \text{(mod p)}$$

$$\equiv g\ \propto_{16}^{1}\ \ldots\ldots \propto_{16}^{m} \propto_{16}^{-m} \propto_{16}^{!s} \qquad \text{(mod p)}$$

$$K``_{16} \equiv g\ \propto_{16}^{1}\ \ldots\ldots \propto_{16}^{m-1} \propto_{16}^{!s} \qquad \text{(mod p)}$$

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

$$\equiv g(\pi_i\alpha_{13})(\pi_i\alpha_{14})(\pi_i\alpha_{15})(\pi_i\alpha_{16}) \qquad \text{(mod p)}$$

$$\equiv g(\pi_i\alpha_{13})(\pi_i\alpha_{14})(\pi_i\alpha_{15})(\pi_{i-1}\alpha_{16}) \qquad \text{(mod p)}$$

$$K``_G \equiv (K`_G)\,\alpha_{16}^{-m} \qquad \text{(mod p)}$$

$$\equiv (PK_{13,16}PK_{9,12}PK_{5,8})^{\alpha_{16}^{1}\cdots\cdots\alpha_{16}^{-m}}\,\alpha_{16}^{m}\alpha_{16}^{!s} \qquad \text{(mod p)}$$

$$\equiv (PK_{13,16}PK_{9,12}PK_{5,8})^{\alpha_{16}^{1}\cdots\cdots\alpha_{16}^{-m}}\,\alpha_{16}^{!s} \qquad \text{(mod p)}$$

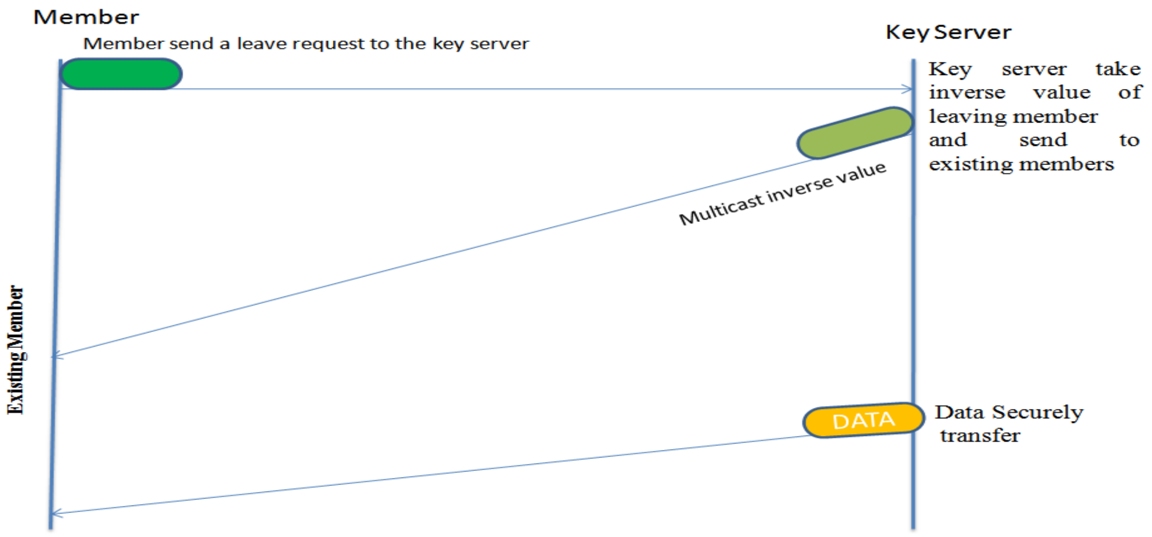
Member
Member send a leave request to the key server
Key Server
Key server take inverse value of leaving member and send to existing members
Multicast inverse value
Existing Member
DATA
Data Securely transfer

Figure 5: Leaving Process of Proposed Approach

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

3 COMPARISON

3.1 Computational Overhead

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

3.1.1 Key Generation Overhead

Key generation overhead at the key server and member node along the path to the root at each join or leave Process and formula for key generation overhead summarize in table 1.

The table 1 shows that 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 6(a) and 6(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.

TABLE 1: Key Generation

Approaches	**Process on Key Server Node**		**Process on Member Node**	
	Join	Leave	Join	Leave
Simple Application	2	1	0	0
LKH	$\log_2 n$	$\log_2 n$	0	0
OFT	$\log_2 n$	$\log_2 n$-1	$\log_2 n$	$\log_2 n$
Our Protocol	$\log_4 \lceil n/m \rceil$	0	0	0

From the figure 6 (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 6(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.

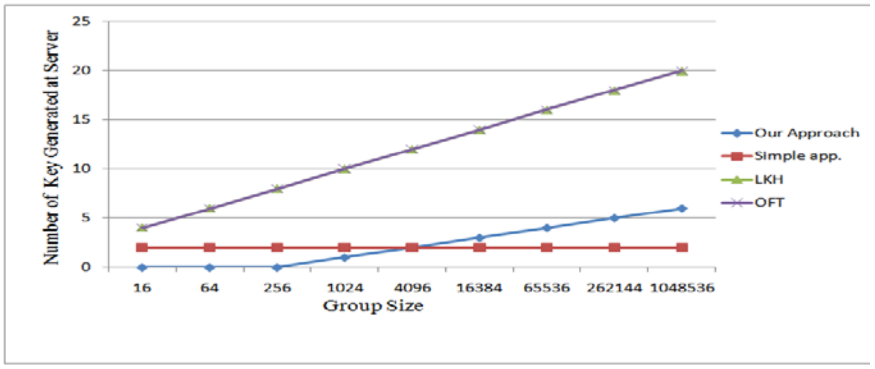

Figure 6(a): Key Generation Overhead at the Key Server during Join Process

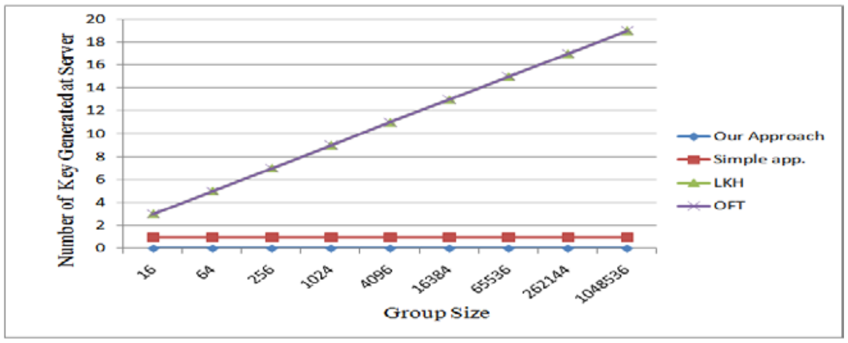

Figure 6 (b): Key Generation Overhead at the Key Server during Leave Process

Figure 7(a) and 7(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.

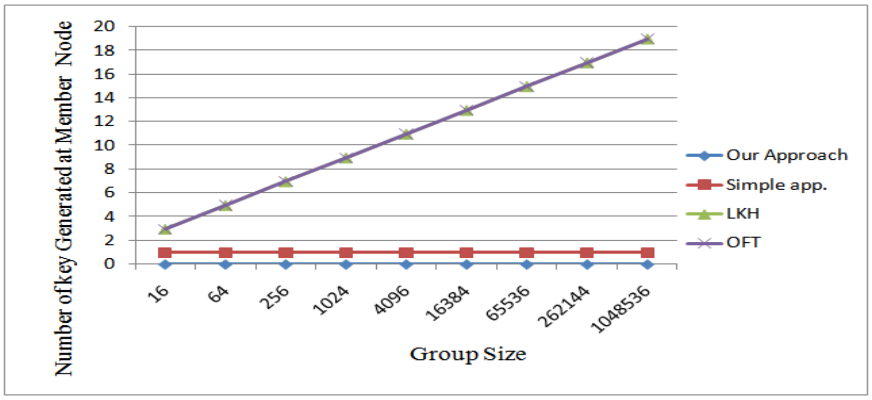

Figure 7 (a): Key Generation Overhead at the Member Node during Join Process

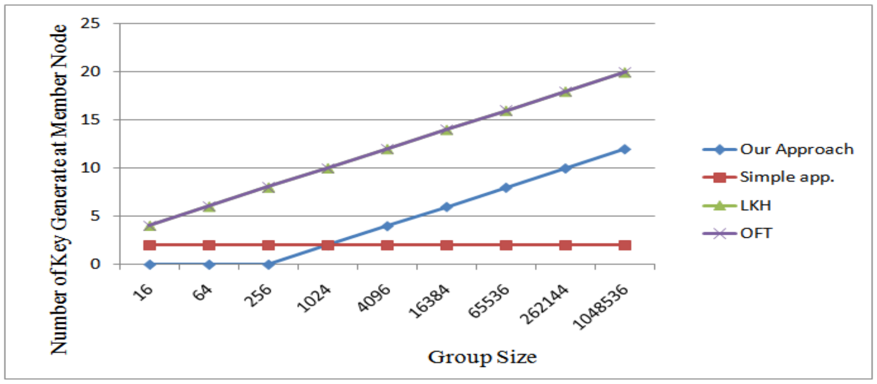

Figure 7 (b): Key Generation Overhead at the Member Node during Leave Process

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.

TABLE 2: Encryption /Decryption Overhead

Approaches	Process on Key Server Node		Process on Member Node	
	Join	Leave	Join	Leave
Simple Application	2	n-1	1	1
LKH	$3\log_2 n$	$2\log_2 n$	$\log_2 n$	$\log_2 n$
OFT	$2\log_2 n+2$	$\log_2 n+1$	$\log_2 n$	2
Our Protocol	$\log_4 \lceil n/m \rceil +2$	$\log_4 \lceil n/m \rceil$	$\log_4 \lceil n/m \rceil$	0

3.1.2 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 as shown in table 2

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.

According to Table 2, 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 (h_{sg}) and small height of group h_g, 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 2, 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.

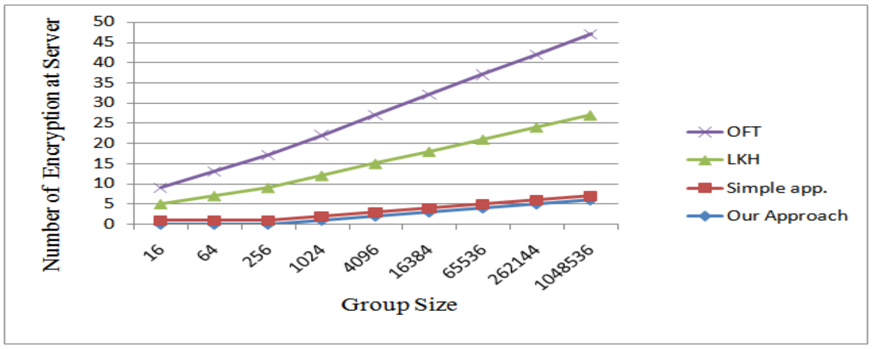

Figure 8 (a): Number of Encryption at the Key Server during Join Process

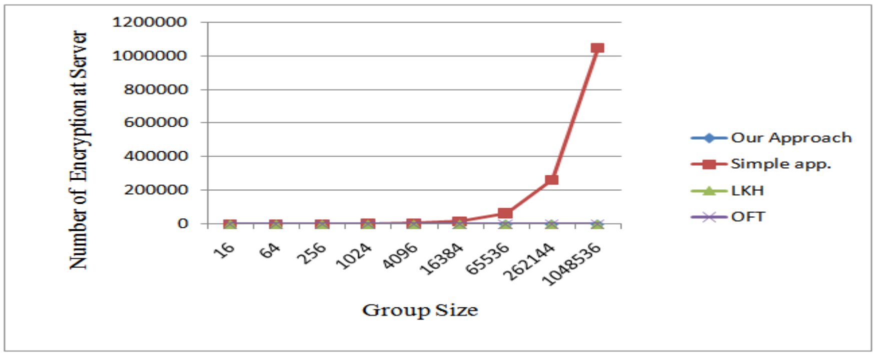

Figure 8 (b): Number of Encryption at the Key Server during Leave Process

Figure 8 (a) and 8(b) show the number of encryption at the key server at join and leave process respectively. In table 2 also shows that 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. Figure 9 (a) and 9(b) show the number of encryption at the member node at join and leave process respectively.

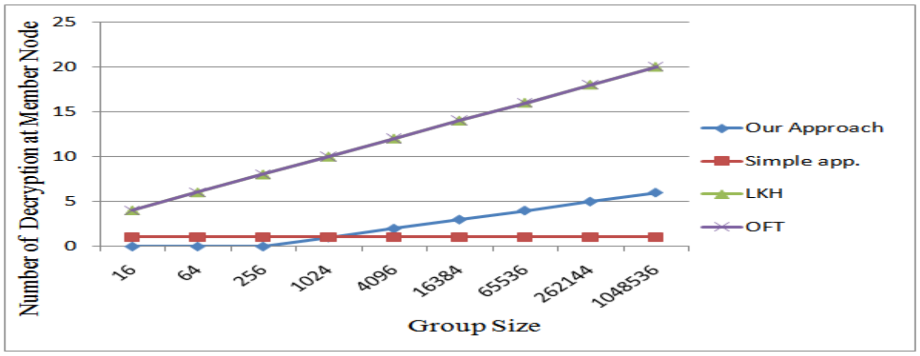

Figure 9 (a): Number of Decryption at the Member Node during Join Process

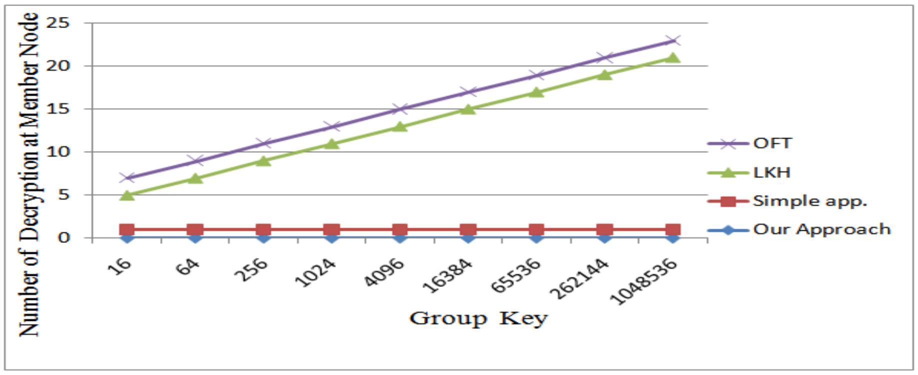

Figure 9 (b): Number of Decryption at the Member Node during Leave Process

3.2 Communication Overhead

Communication overhead at join and leave for multicast communication as shown in table 3. As described above, this is measured by the number of transmitted control messages. Figure 10(a) and 10(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.

TABLE 3: Communication Overhead

Approaches	Process on Key Server Node	
	Join	Leave
Simple Application	1	n-1
LKH	$2\log_2 n-1$	$2\log_2 n$
OFT	$\log_2 n+1$	$\log_2 n+1$
Our Protocol	1	1

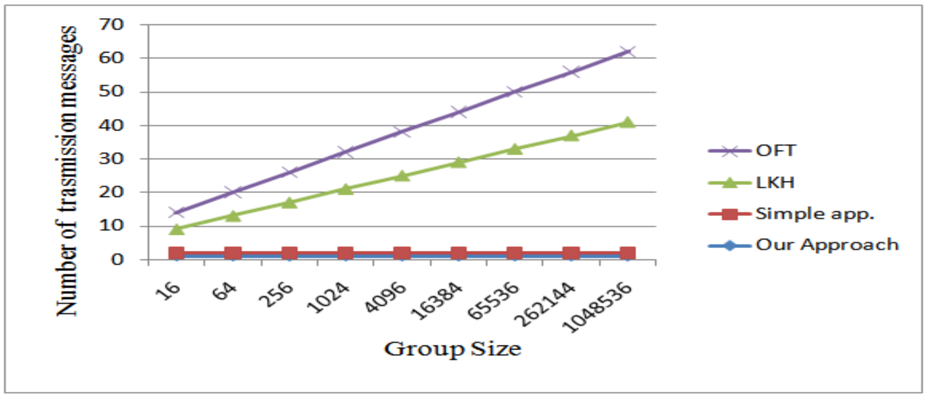

Figure 10 (a): Communication Overhead at Server during Join Process

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.

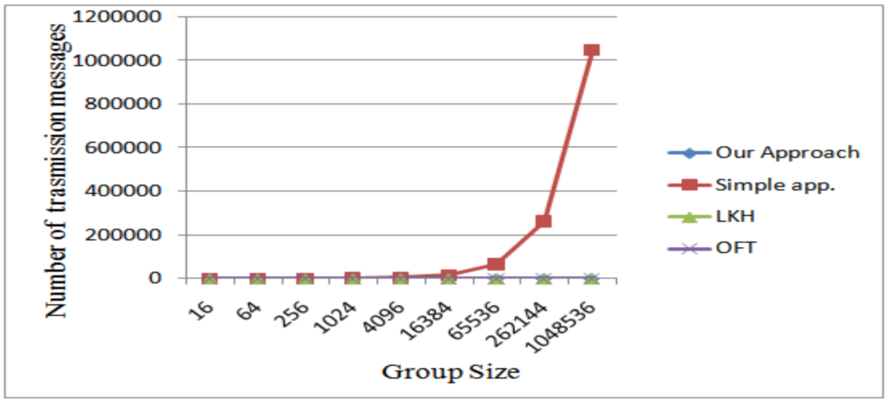

Figure 10 (b): Communication Overhead at Server during Leave Process

TABLE 3: Message Size

Approaches	**Process on Key Server Node**	
	Join	Leave
Simple Application	1	N
LKH	$2\log_2 n$-1	$2\log_2 n$
OFT	$\log_2 n$+1	$\log_2 n$+1
Our Protocol	$\log_4 \lceil n/m \rceil$ +1	$\log_4 \lceil n/m \rceil$

3.3 Rekeying Message Size

Message size is given by the number of keys included in one multicast message and table 3 shows messages size for join and leave. At join, the message size of LKH is the largest, and OFT, our protocol and Simple App. follow in this order.. At leave, LKH have the largest message size, the next is OFT, and our protocol and Simple App. have the smallest size. By considering the communication overhead (the number of transmitted messages) and the message size, our protocol supersedes the other approaches. Figure 11(a) and 11(b) illustrate some numerical results for join and leave, respectively.

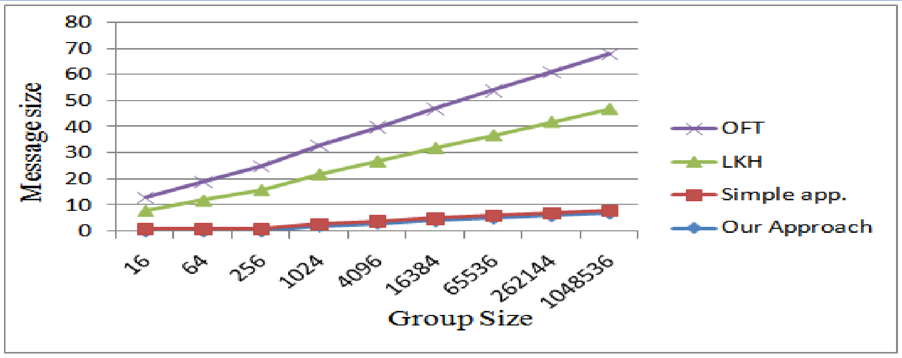

Figure 11 (a): Message Size at Server during Join Process

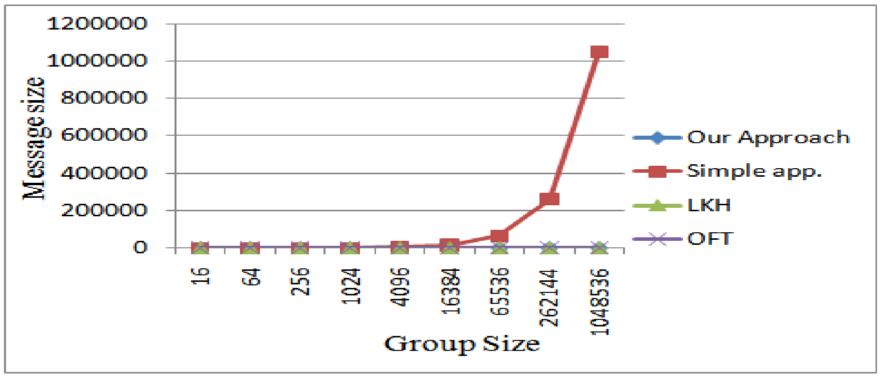

Figure 11 (b): Message Size at Server during Leave Process

TABLE 4: Key Storage Overhead

Approaches	Key Storage on server	Key Storage on member
Simple Application	1	N
LKH	2n	$\log_2 n+1$
OFT	2n	$2\log_2 n+1$
Our Protocol	$\ulcorner 4/3 \urcorner n$	$\log_4 \ulcorner n/m \urcorner +m-1$

3.4 Storage Overhead

Key storage overhead [36] at the key server and a member node as shown in table 4, First of all, the number of keys in a m-ary protocols is equal to $\ulcorner \frac{d}{d-1} \urcorner$ n, when n is the number of members. Therefore, the key storage of our protocol is $\ulcorner 4/3 \urcorner$ n. In our protocol, the number of nodes is equal to 2 $\ulcorner n/m \urcorner$ -1 which is the sum of $\ulcorner n/m \urcorner$ subgroups and $\ulcorner n/m \urcorner$ - 1 intermediate node. Next, according to the formulas, the key server stores fewer keys for Simple App. Meanwhile the key server stores fewer keys for our approach because of its short height compared to LKH and OFT. Figure 12(a) and 12(b) illustrate numerical results for storage overhead at server and member node.

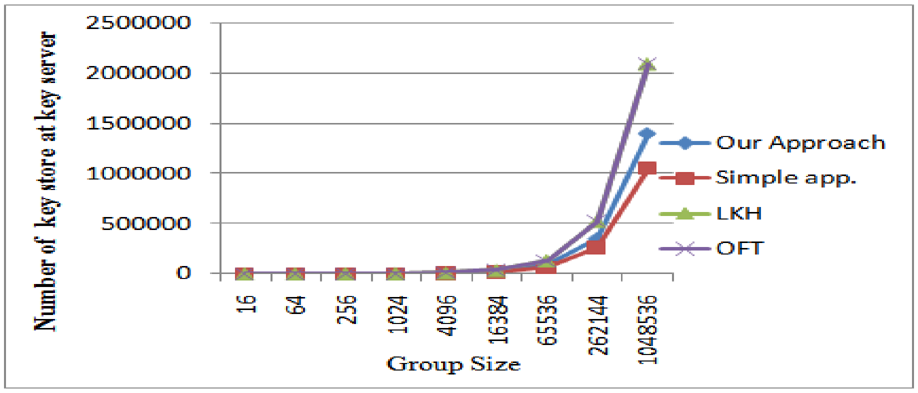

Figure 12(a): Storage Overhead at Server

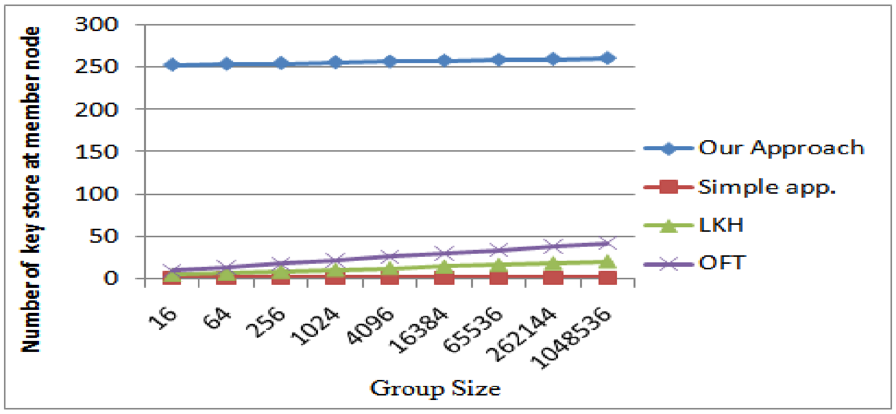

Figure 12 (b): Storage Overhead at Member Node

At a member node, the storage overhead is proportional to the height of the key tree. Table 4, shows that a member node in Simple App. stores only 2 keys which is the smallest storage of all. In OFT, because a member node stores the node keys as well as the hash values of them, the storage overhead gets larger compared to LKH. The storage size in our protocol is the sum of the $\log_4 \ulcorner n/m \urcorner$ node keys and m-1, the secret values of the other members in that subgroup. These comparisons show that when n > 2m, the storage size of our protocol for a member node is large compared with the other protocols, we believe that it is tolerable in most applications. Moreover, we consider the leave operation overhead at the key server to be the most important quantity. Key storage size and the overall members computations are of lesser concern. As shown before, a protocol like Simple App. cannot be scalable for large groups at leave. Consequently, reducing the computational and communication overhead at the key server is the most important factor at leave for re-keying.

4 CONCLUSION

Multicast is the delivery of a message or information to a group of destination systems simultaneously in a single transmission from the source. In the multicast different type of problems occur during communication. One of the problem of multicast is the security. 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. Our approach divides a group of n members into subgroups and size of each subgroup is m members. Each subgroup assign to leaf node of the key tree. When n>2m, we have $\lceil n/m \rceil$ subgroups and height of the logical key tree will be $\log_4 \lceil n/m \rceil$.

We are using inverse values and pre-computation function. The purpose of inverse value is to assign member secret to each member, in order to update the keys efficiently at leave, the keys calculated by the members rather than delivered by key server. Pre-computation function PK will minimize the computation cost during join or leave process. After join or leave process key server re-calculates the key and that key delivered to the members, during re-keying calculation pre-computation function improve the performance of the key server. Each updated keys (group key, node keys and subgroup key) are encrypted by the corresponding previous key at join and transmitted by multicast. However at a leave, the inverse value of the leaving member is sent to remaining members encrypted by each sibling key of the path set and is delivered by multicast. The performance of the proposed protocol is compared with that of the Simple Approach, 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]. R. Srinivasan, V. Vaidehi, R. Rajaraman, S. Kanagaraj, R. Chidambaram Kalimuthu, and R. Dharmaraj" Secure Group Key Management Scheme for Multicast Networks", International Journal of Network Security, Vol.11, No.1, PP.33-38, July 2010.

[2]. S.Jabeenbegum , Dr.T.Purusothaman , 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, VOL.10 No.2, February 2010.

[3]. Chung Kei Wong, Mohamed Gouda, and Simon S. Lam,"Secure Group Communications Using Key Graphs",IEEE/ACM Transactions on Networking, Vol. 8, NO. 1,February 2000.

[4]. Shanyu Zheng, David Manz, Jim Alves-Foss, "A Communication Computation Efficient Group Key Algorithm for Large and Dynamic Groups", Elsevier, Computer Networks, March 2006.

[5]. Marimuthu Rajaram and Thilagavathy Dorairaj Suresh" An Interval-based Contributory Key Agreement" International Journal of Network Security, Vol.13, No.2, PP.92-97, Sept. 2011

[6]. Paul Judge and Mostafa Ammar," Security Issues and Solutions in Multicast Content Distribution: A Survey", IEEE Network, January/February 2003.

[7]. Yacine Challal, Hamida Seba," Group Key Management Protocols: A Novel Taxonomy", International Journal of Information Technology Volume 2 Number 1 2005 Issn: 1305-2403.

[8]. Donghyun Choi, Sungjin Lee, Dongho Won, Seungjoo Kim" Efficient Secure Group Communications for SCADA", IEEE Transactions On PoAuthorr Delivery, Vol. 25, No. 2, April 2010.

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

[10]. Imane Aly Saroit, Said Fathy El-Zoghdy, and Mostafa Matar," A Scalable and Distributed Security Protocol for Multicast Communications", International Journal of Network Security, Vol.12, No.2, PP.61-74, Mar. 2011.

[11]. Said Gharout, Yacine Challal, and Abdelmadjid Bouabdallah" Scalable Delay-constrained Multicast Group Key Management", International Journal of Network Security, Vol.7, No.2, PP.142-156, Sept. 2008.

[12]. T. Hardjono and B. Cain, "Key Establishment for IGMP Authentication in IP Multicast," IEEE ECUMN, CREF, Colmar, France, 2000.

[13]. A. Ballardie and J. Crowcroft,"Multicast-Specific Security Threats and Countermeasures, " Proc. ISOC Symp. Network and Distributed System Security, San Diego, CA, pp. 2-16, Feb. 1995.

[14]. P. Q. Judge and M. H. Ammar, "Gothic: Group Access Control Architecture for Secure Multicast and Anycast," IEEE INFOCOM, July 2002.

[15]. R. Gennaro and P. Rohatgi, "How to Sign Digital Streams," LNCS, vol. 1294, 1997.

[16]. C. Wong and S. Lam, "Digital Signatures for Flows and Multicasts," IEEE/ACM Transaction Network, vol. 7, 1999.

[17]. P. Golle and N. Modadugu, "Authenticating Streamed Data in the Presence of Random Packet Loss," Network and Distributed System Security Symp., 2001.

[18]. P. Rohatgi,“A Compact and Fast Hybrid Signature Scheme for Multicast Packet Authentication,” ACM Conference Computer and Communication Security, Nov. 1999.

[19]. H. Chu, L. Qiao, and K. Nahrstedt, “A Secure Multicast Protocol with Copyright Protection,” Proc. IS&T/SPIE’s Symp. Electronic Imaging: Science and Technoogy, Jan. 1999.

[20]. I. Brown, C. Perkins, and J. Crowcroft, “Watercasting: Distributed Watermarking of Multicast Media,” Networked Group Communication ‘99, Pisa, Italy, pp. 286-300, Nov. 1999

[21]. T. Wu and S. Wu, “Selective Encryption and Watermarking of mpeg Video,” technical report, NC State Univ.

[22]. P. Q. Judge and M. H. Ammar, “WHIM: Watermarking Multicast Video with a Hierarchy of Intermediaries,” Proc. NOSSDAV, Chapel Hill, NC, June 2000.

[23]. D. Wallner, E. Harder and R. Agee,” Key Management for Multicast: Issues and Architecture”. National Security Agency, June 1999. RFC2627.

[24]. Chung Kei Wong, Mohamed Gouda, and Simon S. Lam,“Secure Group Communications Using Key Graphs”,IEEE/ACM Transactions on Networking, Vol. 8, NO. 1, February 2000.

[25]. Shanyu Zheng, David Manz, Jim Alves-Foss,“A communication computation efficient group key algorithm for large and dynamic groups” Elsevier, Computer Networks, Volume 51, Issue 1, Pages 69-93, 17 January 2007.

[26]. Riham Abdellatif, Heba K. Aslan, and Salwa H. Elramly” New Real Time Multicast Authentication Protocol” International Journal of Network Security, Vol.12, No.1, PP.13-20, Jan. 2011.

[27]. Mohamed M. Nasreldin Rasslan, Yasser H. Dakroury, and Heba K. Aslan” A New Secure Multicast Key Distribution Protocol Using Combinatorial Boolean Approach” International Journal of Network Security, Vol.8, No.1, PP.75–89, Jan. 2009.

[28]. Yang Richard Yang, X. Steve Li, X. Brian Zhang, Simon S. Lam” Reliable Group Rekeying: A Performance Analysis” SIGCOMM '01**:** Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications, August 2001.

[29]. Alireza Nemaney Pour , Kazuya Kumekawa, Toshihiko Kato, Shuichi Itoh” A hierarchical group key management scheme for secure multicast increasing efficiency of key distribution in leave operation” Elsevier, Computer Networks, Volume 51, Issue 17, Pages 4727-47435 December 2007.

[30]. Yacine Challal, Hatem Bettahar, Abdelmadjid Bouabdallah” SAKM: A Scalable and Adaptive Key Management Approach for Multicast Communications” ACM SIGCOMM Computer Communications Review, Volume 34, Number 2: April 2004.

[31]. David Manz, Jim Alves-Foss and Shanyu Zheng” Network Simulation of Group Key Management Protocols” Journal of Information Assurance and Security, Volume 3, Issue 1, 67-79, january2008.

[32]. Sandro Rafaeli and David Hutchison” A Survey of Key Management for Secure Group Communication” ACM Computing Surveys, September2003.

[33]. Patrick P. C. Lee, John C. S. Lui and David K. Y. Yau" Distributed Collaborative Key Agreement and Authentication Protocols for Dynamic Peer Groups" IEEE/ACM Transactions on Networking, Vol. 14, No. 2, April 2006.

[34]. Yan (Lindsay) Sun, K. J. Ray Liu" Hierarchical Group Access Control for Secure Multicast Communications" IEEE/ACM Transactions on Networking, Vol. 15, No. 6, December 2007.

[35]. Kin-Ching Chan and S.-H. Gary Chan," Key Management Approaches to Offer Data Confidentiality for Secure Multicast" IEEE Network, September/October 2003.

[36]. Dong-Hyun Je, Jun-Sik Lee , Yongsuk Park , Seung-Woo Seo" Computation-and-storage-efficient key tree management protocol for secure multicast communications," ELSEVIER, Computer Communications ,vol.33, Issue 2, pages 136-148,Febuary 2010.

[37]. Xiaoyan Chen, Bobby N.W. Ma, Cungang Yang" M-CLIQUES: Modified CLIQUES key agreement for secure multicast," Computers & Security, Volume 26, Issue 3, Pages 238-245, May 2007.

[38]. Yu Fang Chung, Hsiu Hui Lee, Feipei Lai, Tzer Shyong Chen," Access control in user hierarchy based on elliptic curve crypto system," Science Direct Information Sciences,vol.178,2008 .

[39]. K.Kumar, Dr.V.Sumathy and J.Nafeesa Begum" Efficient Region-Based Group Key Agreement Protocol for Ad Hoc Networks using Elliptic Curve Cryptography," Advance Computing Conference, IACC2009. IEEE International, March 2009.

[40]. A. Ballardie. "Scalable Multicast Key Distribution," May 1996. RFC 1949.

[41]. B. DeCleene, L. Dondeti, S. Griffin, T. Hardjono, D. Kiwior, J. Kurose, D. Towsley, S. Vasudevan, and C. Zhang. "Secure group communications for wireless networks". MILCOM, June 2001.

[42]. T. Hardjono, B. Cain, and I. Monga." Intra-domain Group Key Management for Multicast Security," IETF Internet draft, September 2000.

[43]. I. Ingemarson, D. Tang, and C. Wong. "A Conference Key Distribution System," IEEE Transactions on Information Theory, 28(5):714–720, September 1982.

[44]. W.Diffie and M.E.Hellman. "New directions in cryptography" IEEE Transactions on Information Theory, IT-22:644–654, November 1976.

[45]. R. Canetti, "Multicast Security: A Taxonomy and Efficient Constructions," IEEE INFOCOM, New York, Mar. 1999.

[46]. A. Perrig, "Efficient and Secure Source Authentication for Multicast," Network and Distributed System Security Symp., Feb. 2001.

[47]. Wee Hock Desmond Ng, Michael Howarth, Zhili Sun, and Haitham Cruickshank,"Dynamic Balanced Key Tree Management for Secure Multicast Communications," IEEE Transactions on Computers, VOL. 56, Page 590 - 605, MAY 2007.

[48]. Haibin Lu," A Novel High-Order Tree for Secure Multicast Key Management," IEEE Transactions on Computers, VOL. 54, Pages: 214 - 224, February 2005.

[49]. Xinliang Zheng, Chin-Tser Huang, Manton Matthews" Chinese Remainder Theorem Based Group Key Management," Proceedings of the 45th annual southeast regional conference, March 2007.

[50]. William Stallings" Cryptography and Network Security Principles and Practices," Fourth Edition, Pages: 592, November 16, 2005.

[51]. D. Cheriton and S. Deering, "Host Groups: A Multicast Extension for Datagram Internetworks," Data Commun. Symp., Sept. 1985, pp. 172-79.

[52]. C.K. Wong, Mohamed Gouda, and Simon S. Lam, "Secure group communications using key graphs", IEEE/ACM Transactions on Networking. 8 (1) (February 2000) 16–30.

[53]. D. Wallner, E. Harder, R. Agee", Key Management for Multicast: Issues and architectures", National Security Agency, RFC2627 (June 1999).

[54]. T. Dierks, E. Rescorla, "The Transport Layer Security (TLS)" Protocol Version 1.1, RFC2346 (April 2006).

[55]. Douglas R. Stinson, "Cryptography Theory and Practice", Second edition, Chapman and Hall/CRC Press, 2002, pp. 155–175.

[56]. C. Rigney, S. Willens, A. Rubens, W. Simpson, "Remote Authentication Dial in User Service (RADIUS)," RFC 2865, June 2000.

[57]. S. Deering "Host Extensions for IP Multicasting," RFC 1112, August 1989.

[58]. D. Waitzman, C. Partridge, S. Deering," Distance Vector Multicast Routing Protocol," RFC 1075, November 1988.

[59]. J. Moy," Multicast Open Shortest Path First (MOSPF), "RFC 1583, march 1994.

[60]. D. Estrin, D. Farinacci, A. Helmy, D. Thaler, S. Deering, M. Handley, V. Jacobson, C. Liu, P. Sharma, L. Wei," Protocol Independent Multicast-Sparse Mode (PIM-SM): Protocol Specification," RFC 2362, June 1998.

[61]. T. Pusateri," Distance Vector Multicast Routing Protocol v3," RFC 1075, August 2000.

[62]. http://www.scalable-networks.com/content/products/qualnet.

[63]. A. Tanenbaum, Computer Networks, Fourth Edition, Prentice Hall, 2009

[64]. Diot, C., et al., ''Deployment Issues for the IP Multicast Service and Architecture,'' IEEE Network, Special Issue on Multicasting, January/February 2000.