An Efficient Scalable Batch-Rekeying Scheme For Secure Multicast Communication Using Multiple Logical Key Trees

Omar Zakaria[1], Aisha-Hassan A.Hashim[1], Wan H. Hassan[2]

[1]Faculty of Engineering, International Islamic University Malaysia
[2]Malaysia - Japan International Institute of Technology (MJIIT),
University Technology Malaysia, Kuala Lumpur

Abstract
Security is vital for many multicast-based application and services. Secure group key management is on of the challenging problems for multicast communication with large number of members. Where for each membership variation the group key must be updated and redistributed to all currently active members only. This causes a higher communication overhead in large size multicast group with high number of users joining or leaving the group. Logical Key Hierarchy which uses the key tree structure is proposed to reduce the communication cost of rekeying procedure and reduces the required number of rekeying messages. Furthermore, batch rekeying is proposed to reduce the rekeying cost by preform rekeying in predefined intervals instead of updating the keys after each join or leave. In this paper, a new scheme based on multiple key trees is proposed. Instead of using only a single key tree multiple key trees are used and at the end of each batch time the algorithm decides which tree will be used to update the keys. This paper shows that utilizing multiple key trees can efficiently decrease the rekeying communication overhead using batch rekeying scheme in tree-based architecture.

Keywords:
Secure multicast, group key management, tree-based system, batch re-keying.

1. Introduction

Group key management play an important role in secure group communication applications where the the data is encrypted using a security key before it is sent to the group members. The session key or group key is distributed to eligible users only, so those only how have the group key can decrypt the received data. The dynamic joining or leaving of users requires frequently updating the group key. This is to achieve a backward and forward secrecy. The backward secrecy guarantees that the joining users can only receive and decrypt the data sent after their joining time and have no accessibility to the data has been sent before that point of time. On the other hand, forward secrecy guarantees that leaving users have no accessibility on the data which is sent after the those users leave the group. Simple rekeying scheme requires that the key server (group controller) sends the new group key to each user one-by-one using the secret key of each individual uses. This simple rekeying scheme is very costly for the groups with large number of users, and the number of rekeying message (communication cost) is equal to the group size. Many schemes has been proposed in literature (such as LKH[1], batch rekeying [2]) to deal with key management in group communication environment. In multicast environment, there are three component a group controller (key server), sending node and multicast group members. Where the group controller is responsible to generate and distribute the security keys to the sender and the active group members. The sending node uses the group key (session key) to encrypt the messages before sending it to the multicast group. The group controller collects the updated information on the multicast group status (the joined and leaved members). Figure 1. shows the secure multicast environment structure.

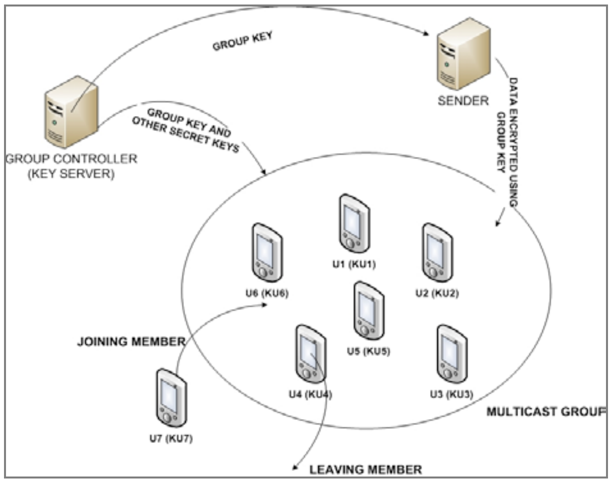

Figure 1. Secure Multicast Environment Components

In this paper, we propose a new key management scheme based on LKH and batch rekeying schemes to reduce the communication overhead. Different from the previous schemes the proposed scheme utilize multiple trees, and select the tree with less rekeying cost at each batch time. From the simulation results, we show that the proposed

Manuscript received November 5, 2014
Manuscript revised November 20, 2014

scheme can optain lower rekeying cost than for balanced a-ary tree with batch rekeying.
The rest of the paper is organized as follows. In section two related works and proposals to reduce the rekeying cost are presented. Section three describe the proposed Multi-logical Tree Key Management scheme (MLT-KM). In section four the proposed scheme is evaluated. Finally, the paper is concluded in section five.

2. Related works

Logical Key Hierarchy (LKH) is most used approach in group key management schemes. It was independently proposed by Wallner et al. [1] and Wong et al. [3]. In LKH the multicast group members are mapped with the leaves of a logical key tree. Each member stores all the keys along the path from its leaf to the tree root. The group key (GK) is located in the root node of the key tree. Intermediate nodes contains the Key Encryption Keys (KEKs), which is is used to encrypt the new keys, while the leafs contain the user security keys (KUs). KUs are shared between the member and the key server before the member join the group. When a member joins/leaves the group, all the keys in his path set need to be changed to a new keys. LKH reduces the rekeying overhead from $O(N)$ to $O(\log(N))$, where N is the size of the group. Figure 2. Shows the LKH key tree (binary key tree) with eight members.

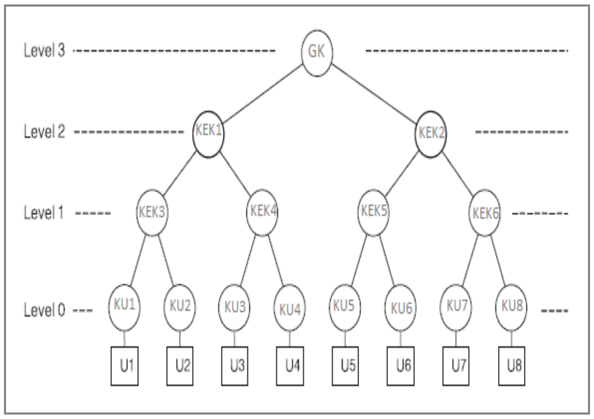

Figure .2 Logical Key Hierarchy Structure

Many proposals aim to optimise the performance of the key-tree structure proposal by achieving a balanced key tree such as [4][5]. Work in [6], proposes a rotation based algorithm to achieve a balanced key tree after each leaving or joining operation, where it can support the cases when the number of leaving members is higher than the joining members.
High number of members leave or join the group by time will invoke a high a mount of rekeying, this degrades the performance of tree-based key management proposal. Therefore batch rekeying was proposed by [2] to deal with this problem. In batch rekeying schemes the group controller does not update the keys immediately after each join/leave operation instead, it waits till the end of fixed time intervals called batch time to perform rekeying process. while using batch rekeying can reduce the rekeying cost by combining the update for many join and leave operations, it has some security limitation where the backward is not well preserved, thus leaving users can still decrypt the data and joined users can not access the data till the end of the batch interval. While it is tolerable if the batching time is relatively small. Batch rekeying scheme assume that the batching time is static, many works [9][10] propose a batch rekeying schemes with dynamic batch rekeying intervals to achieve a trade off between the rekeying cost and the data confidentiality.
Work of [7], reduces the overhead of joining operation using one-way hash function with node coding. Using a binary logical tree structure, when new members join the multicast group, the key server sends the new group key with a unique code to the new members, using the new group key and the node code new members calculate all required middle keys. The remaining members compute the new group key locally by applying one-way hash function to the previous group key.
In [8], n-ary tree structure is used. Their key management scheme reduces the rekeying messages in leaving operation. The number of rekeying messages is paced on the leaving node positions. Members in each subgroup are numbered from 1 to n (the degree of the tree) , members belong to different subgroups is assigned the same key if they are assigned the same number. After each batch time intermediate keys are updated by exoring the old key with the new group key.
Secure Group Key Management Scheme is proposed in [11], their proposal reduces the number of rekeying messages using chinese remainder theorem which combined with LKH. The multicast gruop is divided into multiple clusters where for each cluster there is a new entity called subgroup controller. The inter-cluster key management is done using LKH, for inter cluster key management Chinese Remainder Theorem scheme is used. the idea is that the subgroup controller uses the public keys of the members with the session key which is received from the group member to generate a secure lock which can be only decrypted by the cluster members to get the session key.
Many LKH-based proposal such as in [12][13][14] reduce the average rekeying overhead by organizing the key tree with respect to the rekeying or leaving probability probabilities of members. In [14], a key management scheme to reduce the member leaving rekeying overhead by utilizing the leaving probability of each member, where the group controller calculate the leaving probability based

on the average staying time of each member. This information can be collected from the member past activity profile. Using leaving probability may not be applicable for users in dynamic environment where users have deferent staying time every time they join the group.

3. The proposed Multi-logical Tree key management (MLT-KM) scheme

The leaving and joining operations of members are producing different overhead. Leaving operation needs more rekeying messages and is considered more costly in term of communication overhead this because the key server must use the secret key of each user to encrypt the new keys to guarantee backward security. While joining operation needs less overhead, this because the affected keys is sent to the existing users first then new keys sent to the newly joined member. In our proposal we consider a tree with fixed height (h), with each node at each level has M children except nodes in level h-2 (leafs nodes parents) have a maximum of N number of children, where:

$$N * number\ of\ nodes\ in\ level\ (h-2) \geq maximum\ expected\ number\ of\ members$$

Figure .2 shows the proposed tree structure with (h= 4, M=3, N), the proposed tree is fall under level-homogeneous key tree structure as the classification proposed by [15].

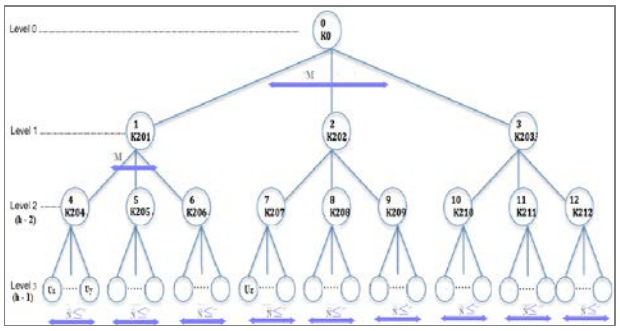

Figure 3. The proposed key tree structure

Table .1 shows a comparision of the rekeing cost of join/leave operation based on the distribution of leaving/joining members over the tree. Where Ux, Uy, Uz are the member positions in the tree (figure .3). The number of rekeying messages is calculated for one or two leaving/joining operations. In different cases when the leaving/joining members are in the same subgroup (Ux, Uy) or different subgroup (Ux, Uz).

Table .1 comparation of different rekeying cost

operation	Users	Number of rekeying messages
Join	Ux	$2+(h-2)*M$
Leave	Ux	$(N-1)+(h-2)*M$
Join	Ux, Uy	$3+(h-2)*M$
Leave	Ux, Uy	$(N-2)+(h-2)*M$
Join	Ux, Uz	$4+(2h-5)*M$
Leave	Ux, Uz	$2*(N-1)+(2h-5)*M$

From Table .1 it can be seen that leaving operation has more rekeying messages than join operation while the worst overhead is when the leaving members are widely distributed from each other (such when Ux and Uy leave the group in the same batch time). The proposed scheme aim to reduce the total cost of leaving operations with a minor increase in the joining operations cost, as a result this leads to reduction in the total communication cost. To control the leaving nodes positions in the tree, multiple key trees are used. At each batch time one tree (primary tree) is selected to distribute the new session key (SK). Since the position of the leaving members determines the number of rekeying messages, the algorithm select the tree which lead to the least number of rekeying messages as primary tree. The KEK of the remaining trees (secondary trees) are then updated by exoring their old keys with the new session key locally on each user node without the need of distribution of the new keys. Let's considesr multiple trees each tree follows the proposed structure and all trees have equal fixed height (h). The root of each tree is the session key (K0) and the leaves contain the private keys of users. Where users distributed randomly on different trees (this to get different distribution of members on different trees subgroups). The intermediate nodes on each key contain the KEK keys (K201, K101, K102,… for first tree; K201, K202,… for the second tree; …. etc). The proposed key management steps after each batch time are described as following:

(1) Add the newly joint member to each tree using joining members algorithm.
(2) Select a primary tree using tree selection algorithm.
(3) In the primary tree all path key's from the root to leafs where the the nodes experiance joining or leaving members are recalculated and the new keys are distributed to the existing member using the same LKH approach.
(4) All key encryption keys of the secondary trees are updated by performing xor operation on the old KEK keys with the new session key to obtain the new keys as following: $K_{new} = k_0 \oplus K_{old}$.

(5) For each secondry tree, distribute the updated keys to the newly joined users only using the same way of LKH.

The new joined members are distributed in different position in different trees to diverse the leaving users distributions within trees. The joining member algorithm is detailed in Figure 4. Let subtree(a, t) donates the subtree which rooted at node ‘a’ in tree ‘t’ and let N determines the maximum number of children (leafs) of each node in level h-2, and let modified(a,t) is a function return 1 if at least one joined or leaved member placed on subtree(a) leaf’s and 0 otherwise. random() is a random function returns values from 0 to 1, $w\ (0 \leq w \leq 1)$ is a weighting parameter where w effects the distribution of new members in the trees by giving more chances to locat newly joining members in subgroups which has experienced join or leave operations, this to reduce the number of rekeying messages due to joining users. Random function is used to distribute the users in random way through different trees.

```
for each j∈set of new joning members do
    for each t∈Trees do
        best=0
        for a ∈level(h-2) in t do
            if number of occupied leafs in subtree(a) < N then
                B = modified(a,t)$*\ w$ + random()
                if B>best
                    bestn=n
                    best=B
        Add j to subtree(bestn)
```

Figure 4. Tree joining member Algorithm

After each batch time the rekeying cost for each tree is calculated and the tree with the least rekeying cost is selected to be a primary tree which will be used to distribute the session key distribution. Tree selection algorithm is presented in Figure 5.

```
maxCost = $\infty$
    for each t∈Trees
        Assign t as a main tree
        Calculate the total rekeying messaging
        (Rekeying Cost)
        if Rekeying Cost<maxCost then
            selectedT = t
            maxCost= Rekeying Cost
    select selectedT as primary tree
```

Figure 5. Tree selection algorithm

To have an example on how the algorithm works, let’s consider a key tree with h = 4, m = 3, n=3 and with maximum number of user 48,where each node on level 2 can have up to 4 children, assume that we use two trees, the first tree is shown in Figure 6 and the second tree in Figure 7. let assume that at the end of a batch time there are three users leaves the group (U1, U4 and U23) and there are three users join the group (U7, U10, U18). The scheme first Add the new members to each tree using joining members algorithm then it selects the primary tree which has less rekeying cost. The required rekeying messages due to joining and leaving members in the tree 1 are as following:

$\{K_{104-new}\}_{KU_2}$, $\{K_{104-new}\}_{KU_3}$
$\{K_{105-new}\}_{KU_5}$, $\{K_{105-new}\}_{KU_6}$
$\{K_{106-new}\}_{K_{106}}$, $\{K_{106-new}\}_{KU_7}$
$\{K_{101-new}\}_{K_{104-new}}$, $\{K_{101-new}\}_{K_{105-new}}$,
$\{K_{101-new}\}_{K_{106-new}}$
$\{K_{107-new}\}_{K_{107}}$, $\{K_{107-new}\}_{KU_{10}}$
$\{K_{109-new}\}_{K_{109}}$, $\{K_{109-new}\}_{KU_{18}}$
$\{K_{102-new}\}_{K_{107-new}}$, $\{K_{102-new}\}_{K_{108}}$,
$\{K_{102-new}\}_{K_{109-new}}$
$\{K_{111-new}\}_{KU_{22}}$, $\{K_{111-new}\}_{KU_{24}}$
$\{K_{103-new}\}_{K_{110}}$, $\{K_{103-new}\}_{K_{111-new}}$, $\{K_{103-new}\}_{K_{112}}$
$\{K_{0-new}\}_{K_{101-new}}$
$\{K_{0-new}\}_{K_{102-new}}$
$\{K_{0-new}\}_{K_{103}}$

Where $\{K_x\}_{K_y}$ denotes that Kx is encrypted using Ky key

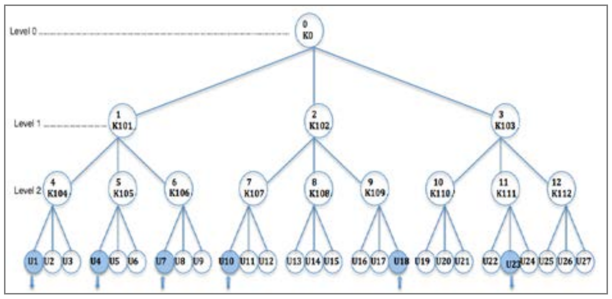

Figure 6 tree-1

While the required rekeying messages in the tree 2 are as following:

$\{K_{204-new}\}_{K_{204}}$, $\{K_{204-new}\}_{KU_{10}}$, $\{K_{204-new}\}_{KU_7}$
$\{K_{201-new}\}_{K_{204}}$, $\{K_{201-new}\}_{K_{205}}$,
$\{K_{201-new}\}_{K_{206-new}}$
$\{K_{206-new}\}_{KU_{25}}$, $\{K_{206-new}\}_{KU_{18}}$
$\{K_{201-new}\}_{K_{204-new}}$, $\{K_{201-new}\}_{K_{205}}$,
$\{K_{201-new}\}_{K_{206-new}}$
$\{K_{208-new}\}_{KU_9}$

$\{K_{202-new}\}_{K_{207}}$, $\{K_{202-new}\}_{K_{208-new}}$,
$\{K_{202-new}\}_{K_{209}}$,
$\{K_{0-new}\}_{K_{201-new}}$
$\{K_{0-new}\}_{K_{202-new}}$
$\{K_{0-new}\}_{K_{203}}$

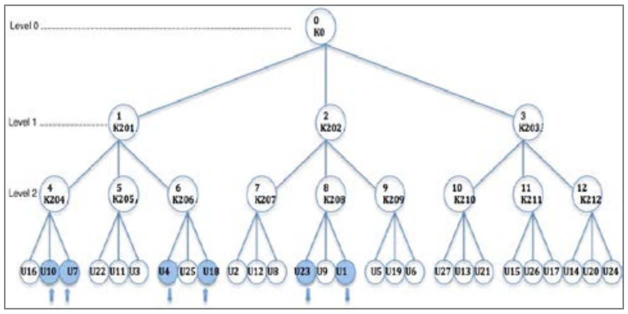

Figure 7. tree-2

It can be seen that using tree-1 to distribute the new session key requires (24) messages while the second tree requires (18) messages, therefore the algorithm selects the second tree to distribute the session key. The key server multicast the encrypted keys related to primary tree (tree 2) only:

$\{K_{204-new}\}_{K_{204}}$, $\{K_{204-new}\}_{KU_{10}}$, $\{K_{204-new}\}_{KU_{7}}$
$\{K_{201-new}\}_{K_{204}}$, $\{K_{201-new}\}_{K_{205}}$,
$\{K_{201-new}\}_{K_{206-new}}$
. . . . . .
. . . . . .
$\{K_{0-new}\}_{K_{201-new}}$
$\{K_{0-new}\}_{K_{202-new}}$
$\{K_{0-new}\}_{K_{203}}$

Using the new group key each user calculates the related new KEK keys of each secondary tree (which is only one tree in this example) by themselves this by xoring the old keys with the new group key as following:

$$K_{101-new} = K_{101-old} \oplus K_{0-new}$$
$$K_{102-new} = K_{102-old} \oplus K_{0-new}$$
$$K_{103-new} = K_{103-old} \oplus K_{0-new}$$
$$K_{104-new} = K_{104-old} \oplus K_{0-new}$$
$$K_{105-new} = K_{105-old} \oplus K_{0-new}$$
$$K_{106-new} = K_{106-old} \oplus K_{0-new}$$
$$K_{107-new} = K_{107-old} \oplus K_{0-new}$$
$$K_{108-new} = K_{108-old} \oplus K_{0-new}$$
$$K_{109-new} = K_{109-old} \oplus K_{0-new}$$
$$K_{110-new} = K_{110-old} \oplus K_{0-new}$$
$$K_{111-new} = K_{111-old} \oplus K_{0-new}$$
$$K_{112-new} = K_{112-old} \oplus K_{0-new}$$

The new KEK keys of the secondary trees then distributed (multicasted) to the new joint users as following:

$\{K_{106-new}\}_{KU_{7}}$, $\{K_{101-new}\}_{K_{106-new}}$
$\{K_{107-new}\}_{KU_{10}}$, $\{K_{102-new}\}_{K_{107-new}}$
$\{K_{109-new}\}_{KU_{18}}$, $\{K_{102-new}\}_{K_{109-new}}$

4. performance evaluation

4.1. Security analysis

The proposed scheme achieves forward and backward secrecy. When a new member joins the group the path keys from the session key (root node) to the new member node in the primary tree is updated. While in the secondary trees the path keys are also updated and the new KEK keys are known for each existing user. The key server then multicasts the updated keys to the newly joint users. The newly joined member will have no information on the old keys or old session key and thus they can not decrypt old messages. Therefore backward secrecy is achieved. In the case of leaving operation all path keys are changed in the primary tree as in LKH procedure, while path keys are changed by xoring the old key with the new session key which is only known by the active members in this way the leaving member will have no information of the new keys. Therefore, forward secrecy is achieved.

4.2. Simulation Results

A simulation is conducted to evaluate the performance of proposed multi-logical tree key management scheme and compare it with a single tree A-ary balanced tree with batch rekeying. At any batch time the number of joining and leaving users are equal, therefore, the total number of members in the group at any time is fixed to 600 member. The prorpsed scheme is compared with 4-aray tree, where for the proposed multi-logical tree key management two trees are used, with height h=5 and M= 4, N=10. In the simulation, the total number of users at each time is 600 users, then at each batch time (x) number of users randomly leave (join) the group. The number of users leave (join) the group (x) after each batch time is varied from 20 to 100. Figure .7 shows the average rekeying messages for our proposal and the 4-ary tree. It shows that the proposed scheme have less rekeying cost in average compared with 4-ary tree when the number of leaving members is increased.

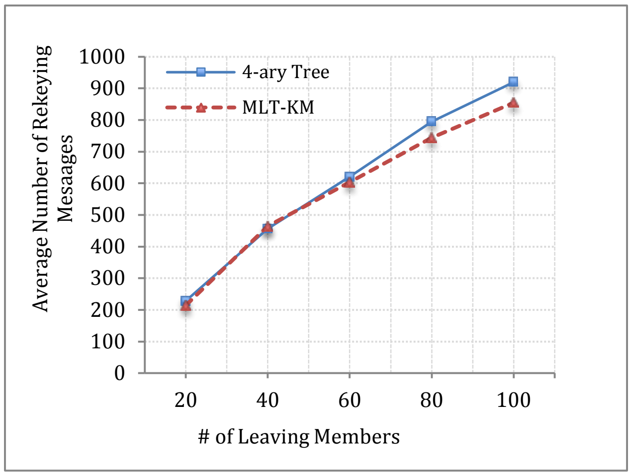

Figure 7. Comparision between MLT-KM and a-ary tree

5. Conclusions

In this paper, a multi-tree key management scheme has been proposed to reduce the rekeying cost for batch rekeying in multicast communication. The proposed scheme reduce the communication cost by controlling the leaving members position this by select the tree in which the leaving members are not widely distribution over the tree. The proposed scheme add some overhead to join operation which is proportional to the number of trees, while in the same time, it effectively reduces the communication cost associated with leaving operation. Many factors affects the performance of the proposed scheme such as the tree structure, the hight and shape of the tree, the tree numbers and the multicast group size. In addition that the joining members distribution algorithm affect the performance of the proposal. Further investigation need to be done to evaluate the impact of each factor on the performance of the scheme.

ACKNOWLEDGMENT

This work is supported by MOSTI under the e-science project (01-01-08-SF0148).

REFERENCES

[1] D. Wallner, E. Harder and R. Agee, "Key Management for Multi- cast: Issues and Architectures," *RFC Key Management for Multicast: Issues and Architectures*, 1999.

[2] X. Li, Y. Yang, M.G. Gouda, and S.S. Lam, "Batch Rekeying for Secure Group Communications", *Proc. 10th Int'l Conf. World Wide Web* , p. 525-534, 2001.

[3] C. K. Wong, M. Gouda, and S.S. Lam, "Secure Group Communications Using Key Graphs," IEEE/ACM Trans. Networking, vol. 8, no. 1, p. 16-30, 2000.

[4] J. Pegueroles and F. Rico-Novella, "Balanced Batch LKH: New Proposal, Implementation and Performance Evaluation" *Proc. Eighth IEEE Int'l Symp. Computers and Comm. (ISCC '03)*, p. 815-820, 2003.

[5] W. Ng, M. Howarth, Z. Sun, and H. Cruickshank, "Dynamic Balanced Key Tree Management for Secure Multicast Communications," *IEEE Trans. Computers*, vol. 56, no. 5, p. 590- 605, 2007.

[6] P. Vijayakumar, S. Bose, A. Kannan, "Rotation based secure multicast key management for batch rekeying operations", *Network Science*, vol. 1, no. 1-4, p. 39-47, 2012.

[7] M. Hajyvahabzadeh, E. Eidkhani, S. Mortazavi and A. Pour, "An efficient group key management protocol using code for key calculation: CKC", Telecommunication Systems, vol. 51, no. 2-3, p. 115-123, 2012.

[8] R. Varalakshmi and V. R. Uthariaraj. "A new secure multicast group key management using gray code". *IEEE-international conference on recent trends in information technology, ICRTIT* 2011, June 3–5, 2011.

[9] J. H. Cho, I. R. Chen and M. Eltoweissy, "On optimal batch rekeying for secure group communications in wireless networks," *Wireless Networks*, vol. 14, no. 6, pp. 915–927, 2008.

[10] D. Je, H. Kim, Y. Choi and S. Seo, "Dynamic Configuration of Batch Rekeying Interval for Secure Multicast Service", International Conference on Computing, Networking and Communications (ICNC), 2014, p.26-30,2014.

[11] E. Munivel, J. Lokesh, "Design of Secure Group Key Man-agement Scheme for Multicast Networks using Number Theory". *International Conference on Computational Intelligence for Modelling Control & Automation 2008, p. 124-129. 2008.*

[12] A. A. Selçuk and D. P. Sidhu, "Probabilistic methods in multicast key management". *Proceedings of the Third International Workshop on Information Security*, 2000.

[13] A. R. Pais and S. Joshi,"A new probabilistic rekeying method for secure multicast groups", *International Journal of Information* Security, vol. 9, no. 4, p. 275-286, 2010.

[14] Y. Park, D. Je, M. Park, and S. Seo. "Efficient Rekeying Framework for Secure Multicast with Diverse-Subscription-Period Mobile Users," *IEEE Trans. Mobile Computing*, Vol. 13, No. 4, p. 783 - 796, 2014.

[15] D. Je, J. S. Lee, Y. Park, and S. Seo, "Computation-and-Stor-age-Efficient Key Tree Management Protocol for Secure Multicast Communications" *Computer Comm.*, vol. 33, no. 2, p. 136-148. 2010.