IJCSE International Journal of Computer Sciences and Engineering Open Access
Research Paper Vol.-7, Issue-3, Mar 2019 E-ISSN: 2347-2693

Analysis of Tree structure for Secure Group Communication Using LKH Approach

Aparna S. Pande[1*], Yashwant V. Joshi[2], Manisha Y. Joshi[3], Lalitkumar Wadhwa[4]

[1]Department of Computer science &Engineering, SGGSIE&T, Nanded, Maharashtra, India
[2] Department of Electronics &telecommunication Engineering, SGGSIE&T, Nanded, Maharashtra, India
[3]Department of Computer science &Engineering, MGMCE, Nanded, Maharashtra, India
[4] Department of Electronics &telecommunication Engineering, NMIET, Pune, Maharashtra, India

[*]*Corresponding Author: aparnaspande10@gmail.com*

DOI: https://doi.org/10.26438/ijcse/v7i3.11301136 | Available online at: www.ijcseonline.org

Accepted: 26/Mar/2019, Published: 31/Mar/2019

***Abstract*—** Logical Key Hierarchy is a scalable and efficient method to achieve logarithmic rekeying cost in secure group communication. In applications like pay per view, video conferencing with multiple rekeying operations, the key tree will be unbalanced and will generate worst case rekeying cost. With each join, leave operation we change group key, as well as update all keys along the key path of join/leave user. Key aspect in secure group communication is maintained balanced key tree and achieving logarithmic rekeying cost. In this paper improvement in Non-split balancing high order tree is proposed. I-NSBHO (improved Non Split Balancing High order tree) with proposed Join user algorithm and leave user algorithm maintains balance of tree and always achieve logarithmic rekeying cost. Our experimental result shows the achieved improvement in rekeying cost of I-NSBHO join and leave operations compared to original NSBHO join and leave operations. With Node pruning I-NSBHO improves join cost and maintains logarithmic Rekeying cost for leave operation

Keywords— *Secure Group Communication, NSBHO Tree, Logical Key Hierarchy, Message cost, Rekeying, Key tree, key path, Logarithmic Rekeying Cost)*

I. INTRODUCTION

In IP multicast, secure group communication is not provided by the network, thus for secure group communication a group key known to all group members is used. With change in group dynamics due to join group or leave group, it is necessary to change the group key and distribute this change in group key to all current group members. This entire process of change and updating of new group key is called as rekeying. This achieves backward and forward secrecy. Backward secrecy is maintained if a user joins the group now, the previous group communication is not accessible to that user. Forward secrecy is achieved, if a user leaves group now, will not be able to get further group communication. In recent years many scalable rekeying approaches are proposed, but among all, LKH and its variants are widely used [1], [3], [9]. In hierarchical tree approach [9], [11] a key tree of current group-users is maintained. In this rooted tree, where leaf node denotes a user and all users are at the same level is built in bottom up manner. Server has key tree structure of current group-users. The server is responsible for authentication of user participating in group communication. Server has authentication information of each user along with its individual key. Server generates and distributes group key to all current group users in group communication.

Figure 1 describes centralized key management and distribution in hierarchical key tree approach. Server handles three types of keys, group-users individual key/user key (UK), subgroup keys (SGK), and group key (GK). Figure 1 show, key operations required in secure group communication at server side and user side [12]. PRNG (pseudorandom number generator) generates user's individual key. Figure 1 describes rekeying module, role of encryption and decryption in applications like pay per view [2].For each user, hierarchical key tree approach has logarithmic storage cost [5]. In hierarchical key tree approach logarithmic rekeying transmissions are required [6]. Figure 1 explains role of server in hierarchical key tree approach. It also describes unicast and multicast messages required for secure group communication at server and user side.

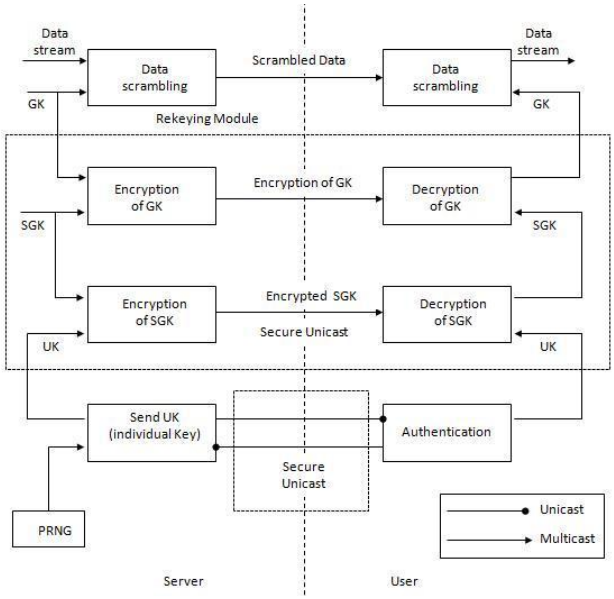

Fig.1 Key management in Hierarchical key tree approach

To evaluate performance of key distribution algorithm, parameters like message cost[2], tree balancing cost, join cost, leave cost, rekeying cost, order, degree, height of tree; confidentiality, secrecy and overall computation cost are used [4],[10]. In this paper among these parameters rekeying cost and overall computation cost are used to compare the performance of NSBHO and I-NSBHO. This paper is organized in sections. Section 2 explains role of tree structure in LKH approach. Section 3 explains NSBHO tree with join and leave operations. Section 4 explains proposed change/improvement in NSBHO tree (I-NSBHO). Section 5 is performance analysis of proposed method. Section 6 is summary/conclusion of the paper.).

II. RELATED WORK

In LKH [11] a rooted tree of keys known as Key tree is constructed in bottom up manner. Server constructs and maintains the current user Key tree. As mentioned in figure 2, Key tree has U-nodes and K-nodes[10]. At leaf, U-node indicates users; all users are at same level. K-nodes indicate either subgroup key or group key. All current group members share group key to participate in group communication. In figure 2, U1, U2, U3, U4 are U-nodes, K1, K2, K3, K4 are their user individual keys. K12, K34 are subgroup keys and K14 is group key. Each user maintains its user key, cur-rent group key and all intermediate keys from its position up to the root of the key tree. The key length from current node to the root is known as key path or external path. User U1 has in its key path keys K1, K12 and K14.

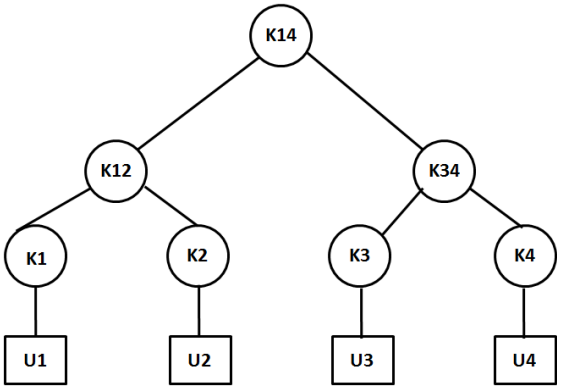

Fig.2. 2-3 Tree *(Key Tree)*

When a user joins the group or leaves the group we change the keys in its key path to maintain forward and backward secrecy [7]. When user U1 leaves the group, then all keys in its key path are changed to maintain forward secrecy. Operations that required maintaining forward secrecy is delete K1, Change K12 and K14. Distribute new (changed) keys to respective users, so multicast new group key K24 to remaining users, unicast K2 to user U2.Figure 3 shows the change in 2-3 key tree when U1 leaves the group[10].

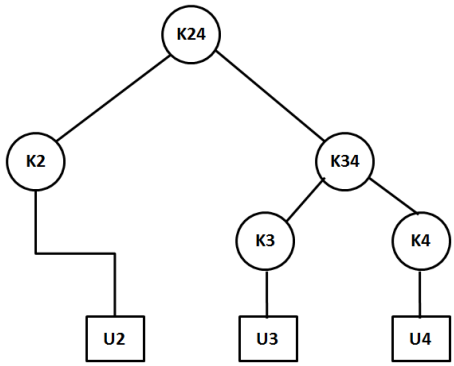

Fig.3. 2-3 Key Tree when U1 leaves the group

If user U5 joins the group, then to maintain backward secrecy, affected keys in its key path are updated. Figure 3 shows updated 2-3 Key Tree when U5 joins the group. In fig 3, K15 is new group key and subgroup key K34 is changed to K35. To update keys, multicasts the new group key to all users and unicast the key path to U5. During join user operation old keys are used for encryption of new keys which is not possible in leave user operation [4], [6]. With this LKH maintains forward secrecy and backward secrecy. LKH is secure from collusion of evicted users [8].

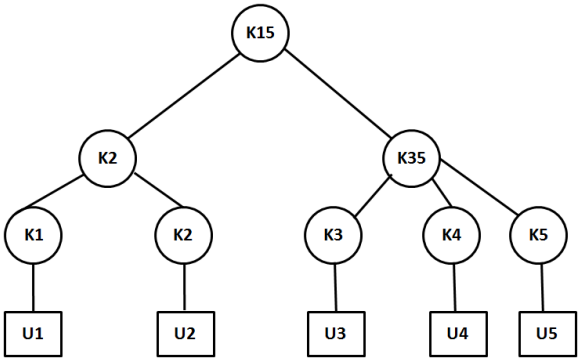

Fig.4. **2-3-4** Tree when U5 joins the group

LKH is efficient and achieves logarithmic rekeying cost if the key tree is remained balanced. LKH has very little control on shape of the key tree. Over a period (after many join and leave operations) 2-3 tree is unbalanced. Balancing tree results in node splitting [1], [3]. Node splitting cost is (m+2) h, where h is height and m is order of tree [3]. This makes 2-3 tree not efficient for long session groups.

III. NON SPLIT BALANCING HIGH ORDER TREE

We assumed here, to have a trusted server which stores group membership information. Whenever, there is a join request from a user to the group, this server with authentication protocol authenticates user. Server maintains access list of all authenticated members. Server distributes group key to group. To keep forward secrecy /backward secrecy, Server updates new group key with change in group dynamics, to ensure only current group members are participating in group communication.

Habin Lu [3] proposed, Non-Split Balancing High-Order tree with one special path. Nodes from special path do not follow standard B-tree properties. Figure 5 shows a NSBHO order 3tree. From [3] an empty tree and tree with only one node (leaf node) is NSBHO order m tree. All users are at same level. In NSBHO order m tree, internal nodes have at most m and at least d nodes [d=m/2]. Root has at least two and at most m nodes. Special path has chain of zi nodes with Z0 as root. z0; z1; . . . ; zk , zi has at least one node and at most d - 1 nodes. Z0 is rightmost child of root.

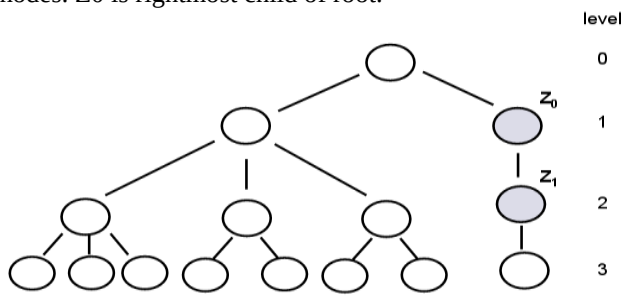

Fig 5 . NSBHO order 3 tree, shaded nodes are in the special path (SP)

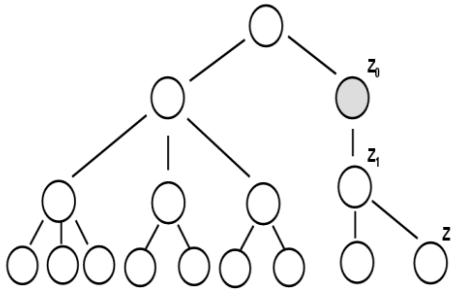

Fig.6. NSBHO order3 tree join operation, join node Z in special path

Algorithm Join (Node M, T)
{ If {T is null, M becomes root, return; }
Node y = add_ node ();
If { y is null, y is new root}
Else make current root as child of y;
For Special Path SP, Attach z as the rightmost child of y; **}**

Algorithm add_node ()
{ If (Internal nodes are full return null ;
Else Return nonfull internal node ;
If {Special path is not full; return the highest level node in SP; }

Fig.7.Algorithm Join M to key tree T

Algorithm Leave (Node M,T)
{
If {M is root, set T=null; return ;
}
Elseif (M is not root && (nodes_of(M)== d-1)
{ get_rich_sibling(M);
Delete M and move a sibling of M to become M
}
if (Nodes_of(M)>1) Merge siblings of M
Else {if M is in SP, call leave (Node M), for parent of M, delete M. }}

Fig.8.Algorithm Leave M from key tree T

Algorithm Join and Leave explains NSBHO join user and leave user operation. In NSBHO tree special path operation takes O(1) time, with special path insertion rekeying cost is reduced. NSBHO tree [3] is a balanced tree so no balancing overhead. NSBHO has same deletion rekeying cost as that of B-tree(2-3 tree) [3],[10]. Table 1 lists rekeying cost for join and leave user operation

IV. IMPROVED NSBHO

In this paper to improve performance of LKH we proposed improvement in NSBHO tree. In NSBHO tree special path operations require O(1) time. If maximum special path operations are performed then overall rekeying cost will be reduced. With this aspect, we proposed modification in NSBHO tree join and leave user operation.

As NSBHO and B-tree (2-3 Tree) have same rekeying cost [3]. From literature survey [2],[3],[8],[10] it is been observed that leave user operation is complex and results in more rekeying cost than join user. I-NSBHO nodes follow all properties of node de-fined for NSBHO [3]. In I-NSBHO we focus on Leave user operation. We set prune flag whenever user leaves the group and we try to fill that position with nodes from special path. When a user leaves group, to maintain forward secrecy we change entire keys along path from that node till root of the tree. So before rekeying, shift a highest level node from special path to that position and then invoke rekeying operation. Algorithm for leave user and join user is explained in fig9 and fig 10 respectively. This change will improve join cost with each leave user operation .With this we always make room in special path to insert/join new node. Special path operation takes O (1) time. This improves average case leave rekeying cost also.

```
Algorithm Leave (Node M,T)
{ If {M is root, set T=null; return ;
}
Elseif (M is not root && (nodes_of(M )== d-1 )
{ Set prune flag ,
If Special path is not empty;
return the highest level node in SP Delete M,
move highest level node from SP to become M;
Else get_rich_sibling(M);
 Delete M and move a sibling of M to become M
}
if (Nodes_of(M)>1) Merge siblings of M
Else {if M is in SP, call leave (Node M), for parent of M,
delete M. }}
```

Fig.9. Algorithm Leave M from key tree T

```
Algorithm Join (Node M, T)
{ If {T is null, M becomes root, return;
 }
If ( Node flag == pruned)
Node y = add_ node ();
If { y is null, y is new root
}
 Else make current root as child of y;
 For Special Path SP, Attach z as the rightmost child of y;
 }
Algorithm add_node ()
 {
If {Special path is not full;
return the highest level node in SP;
If ( Internal nodes are full return null
Else Return nonfull internal node }
```

Fig.10.Algorithm join M to key tree T

With this proposed join and leave algorithm, we tried to maintain shape of key tree. We restrict width of key tree to improve performance. With node pruning, node positions with pruned flag are reused to maintain shape of key tree. As mentioned in fig 10, when a user leaves the group set pruned flag for that node and move highest level node from SP to that position. In fig 9 while joining a new user, pruned positions are reused to maintain shape of key tree. I-NSBHO maintains balance of key tree after join/leave operation ensuring logarithmic rekeying cost. Complexity of proposed join/ leave algorithm is O (h+logn), where h is height of key tree.

V. PERFORMANCE ANALYSIS OF THE PROPOSED METHOD

To analyze performance of proposed I-NSBHO method we used parameters computational time and message cost vs number of join and leave operations. The efficiency of I-NSBHO method is evaluated by Ns2 and Java script. Our results shows achieved improvement in I-NSBHO compared to NSBHO. I-NSBHO grows slower with increased group size and number of Join/leave operations. We restrict increment in width of key tree, so at each level there is scope for additional new user's accommodation with increment in height/level. Table 1 shows comparison of 2-3 tree, NSBHO tree and I-NSBHO tree. In join/leave operation, key tree of 8, 16, 32, 64,128 users are used. Each figure shows, number of keys generated and total computation time required performing join/leave operation. Change in colour shows key path of operation and nodes participated in rekeying.

Figure 11, Figure 13 and figure 15 shows I-NSBHO join operation with key tree of 8user, 16user and 32 users respectively.

Figure 12, Figure 14, figure 16 shows I-NSBHO leave operation with key tree of 8user, 16user and 32 users respectively. Figure 17 shows comparison of overall computation time required for NSBHO & 2-3 tree LKH join operation with 128 user key tree. Figure 18 shows comparison of overall computation time required for 2-3 tree LKH and NSBHO leave operation with 128 user key tree. Figure 19 and Figure 20 shows comparison of NSBHO and I-NSBHO join and leave operation with 128 user key tree. Our

experimental results showed achieved improvement in I-NSBHO join and leave operation.

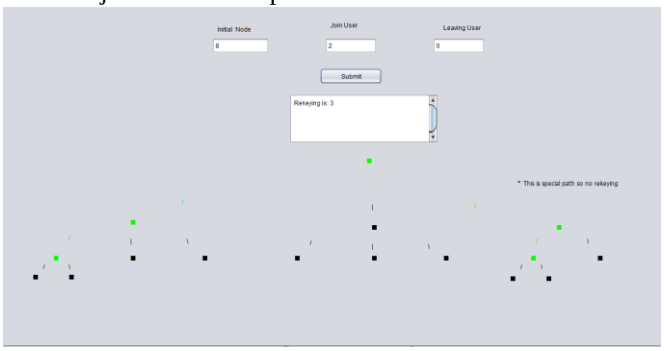

Fig.11.I-NSBHO 8 user key tree join operation

Fig.12. I-NSBHO 8 user key tree Leave operation, **when users 1,2 leave the group**

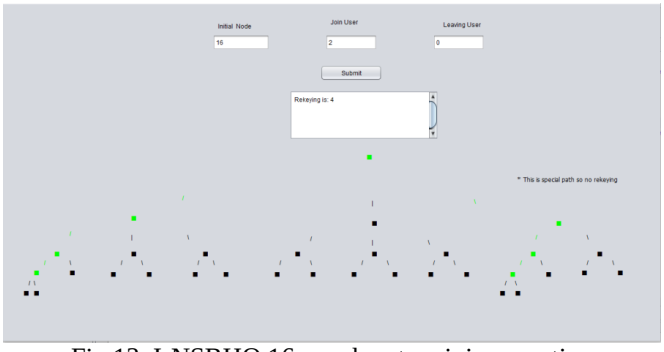

Fig.13. I-NSBHO 16 user key tree join operation

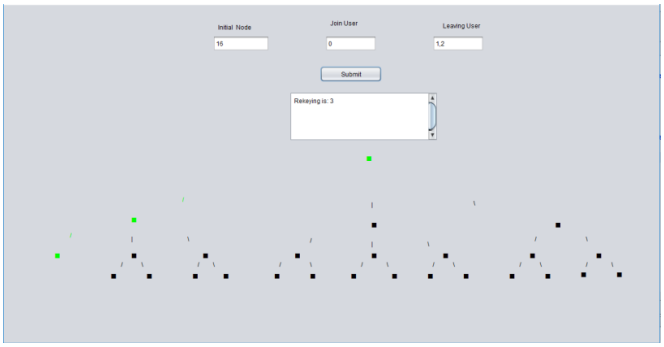

Fig.14. I-NSBHO16 user key tree leave operation, when users 1,2 leave the group

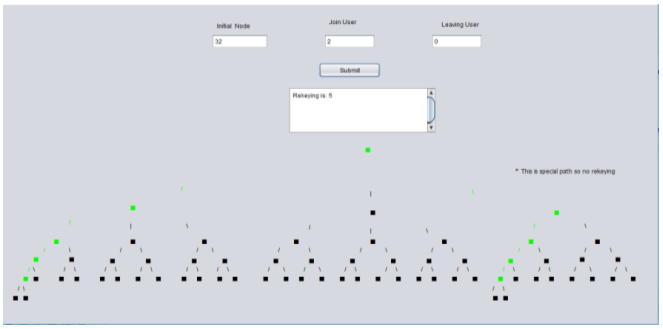

Fig.15. I-NSBHO 32 user key tree join operation

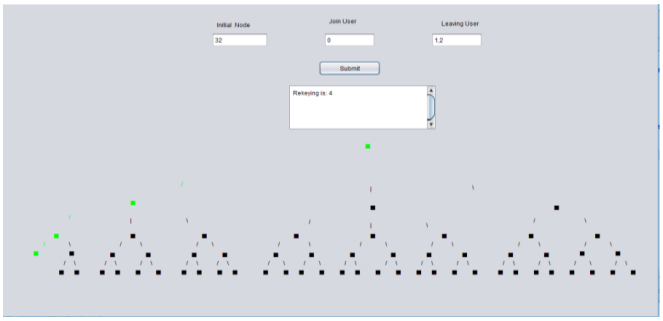

Fig.16. I-NSBHO 32 user key tree leave operation, when users 1,2 leave the group

Table 1 Overall comparison of 2-3 tree, NSBHO and I-NSBHO

Parameter	2-3 tree	NSBHO	I-NSBHO
Method	Centralized , bottom up	Centralized, bottom up	Centralized, bottom up
Tree	2,3,4 tree	NSBHO Tree	NSBHO Tree
Tree Balancing method	Required, Node splitting	Not required	Not required
Focused on	Length of tree	Special Path	Special Path, node pruning
Node pruning	No	No	yes
Skewness	yes	No	No
Suitable for Application	Short session groups	Long session large groups.	Long session large groups.
Leave cost Best case	d(h-1)+2	1	1
Leave cost Worst case	d-1+mh	d-1+mh	h+logn
Join cost Best case	2h	h+1	h+1

Notations used in paper And Tables

- N is number of group members.
- d is depth of tree
- m is order of tree
- h is height of tree
- PRNG is pseudo random number generator

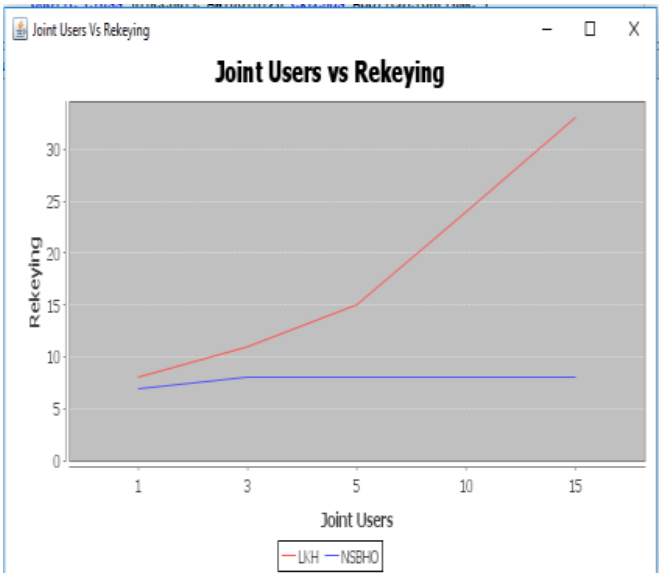

Fig.17. I-NSBHO 128 user key tree join operation

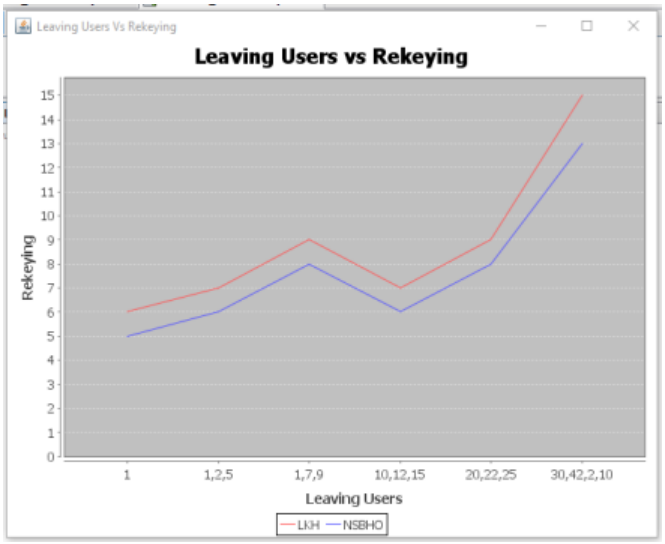

Fig.18. I-NSBHO 128 user key tree Leave operation

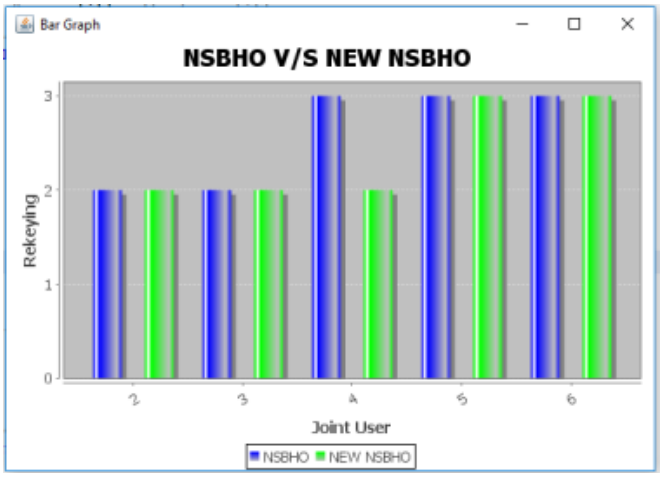

Fig.19 NSBHO and I-NSBHO join operation 128 User Key Tree

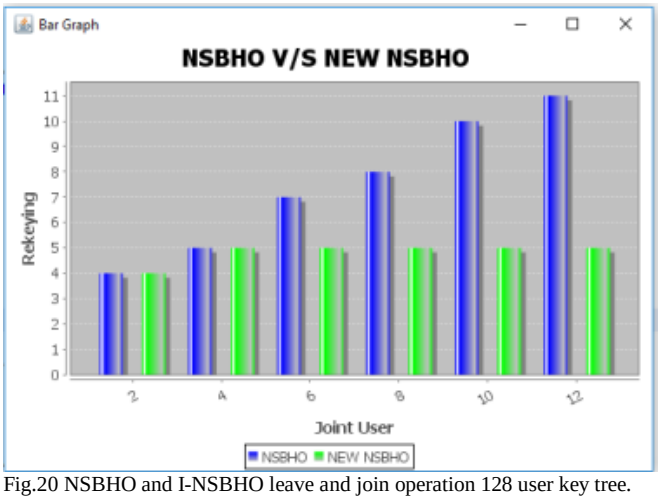

Fig.20 NSBHO and I-NSBHO leave and join operation 128 user key tree.

VI. CONCLUSION

I-NSBHO uses proposed join and leave algorithm, maintains shape of key tree and with node pruning improves performance. I-NSBHO join and leave algorithm reduced rekey overhead. I-NSBHO achieves logarithmic rekeying cost for long session groups. I-NSBHO maintains shape of key tree so extra tree balancing effort is not required. In LKH, 2-3 key tree balancing results in node splitting [3], where as in I-NSBHO with node pruning positions are reused. I-NSBHO 8, 16, 64 and 128 user key tree is used for analysis. Table 1 gives comparison of method 2-3 tree, NSBHO tree and I-NSBHO Tree. In key management role of tree structure is prime, with our experimental results we proved NSBHO tree is far better than 2-3 tree [3]. In [10] author proved 2-3 tree is better compared to counter tree. I-NSBHO improves average case rekeying cost. In future I-NSBHO tree can be used in distributed approach

REFERENCES

[1] Takahito Sakamoto, Takashi Tsuji, Yuichi Kaji. "Group key rekeying using the LKH tech-nique and the Huffman algorithm" , 2008 International Symposium on Information Theory and Its Applications, 2008

[2] Kuei-Yi Chou, Yi-Ruei Chen, Wen-Guey Tzeng. "An efficient and secure group key man-agement scheme supporting frequent key updates on Pay-TV systems" , 2011 13th Asia-Pacific Network Operations and Management Symposium, 2011

[3] H. Lu. "A novel high-order tree for secure multicast key management" , IEEE Transactions on Computers, 2/2005

[4] H. Zhou, M. Zheng and T. Wang, "A Novel Group Key scheme for MANETS", Advanced in control Engineering and Information Science, Procedia Engineering, IJSIA, 2011, pp. 3388-3395.

[5] S. Zhao, R. Kent and A. Aggarwal, "A Key Management and secure routing integrated framework for Ad-hoc Networks", Ad-hoc Networks, Elsevier, vol-11, 2013, pp. 1046-1061

[6] Hanatani, Yoshikazu, et al. Secure Multicast Group Management and Key Distribution in IEEE 802.21. Security Standardisation Research Springer International Publishing. 2016, pp. 227-243.

[7] S.K. Hafizul Islam, G.P. Biswas A pairing-free identity-based two-party authenticated key agreement protocol for secure and efficient communication J. King Saud Univ.-Comput. Inf. Sci., 29 (1) (2017), pp. 63-73

[8] Vinod Kumar, Rajendra Kumar, S.K. Pandey, "A computationally efficient centralized group key distribution protocol for secure multicast communications based upon RSA pub-lic key cryptosystem", Journal of King Saud University - Computer and Information Sciences, 2018.

[9] Aparna S. Pande Y. V. Joshi, Manisha Y. Joshi, "Analysis on Logical Key Hierarchy and Variants for Secure Group Communication", ICCASP 2018, Book chapter, Springer-Atalntis, AISR, ISSN 1951-6851.

[10] J. Goshi and R.E. Ladner, "Algorithms for Dynamic Multicast Key Distribution Trees," Proc ACM Symp. Principles of Distributed Computing (PODC 2003), 2003.

[11] C. K. Wong, M. Gauda and S. S. Lam, "Secure Group Communications Using Key Graphs," IEEE/ACM Transactions on Networking, Vol.8, no.1, pp.16-30, 2000.

[12] P.Vijayakumar, S.Bose, A.Kannan, "Centralized key distribution protocol using the great-est common divisor method", Computers and Mathematics with Applications , volume 65 1360–1368, 2013

Authors Profile

Aparna S. Pande pursed Bachelor of computer Science and Engineering from Swami Ramanand Tirth marathwada University of Nanded, Maharashtra in 2003 and Master of computer Science and Engineering from Swami Ramanand Tirth marathwada University of Nanded, Maharashtra in year 2010. She is currently pursuing Ph.D. and currently working as Assistant Professor in Department of Information Technology, Nutan Maharashtra Institute of Engineering and Technology , University of Pune, Maharashtra since 2008. She is a member of IEEE Pune section, a life member of the ISTE since 2004, IAENG since 2018. She has published 3 research papers in reputed international journals and conferences including Springer and it's also available online. Her main research work focuses on Network Security and Privacy, Big Data Analytics. She has 10 years of teaching experience and 5 years of Research Experience.

Yashwant V. Joshi completed Bachelor of Electronics Engineering in 1986, Masters of Electronics Engineering in 1991 (both from SGGS Institute of Engineering and Technology, Vishnupuri, Nanded) and a Ph.D. in 1998 (from IIT, Delhi). He is member of IEEE, Fellow of Institution of Engineers (India), Fellow of IETE (India) and a life member of ISTE.He is Director of SGGSIET from april 2018. His subjects of interest include Signals and systems, Digital Signal Processing, Adaptive Signal Processing, Modern Digital Design using Verilog, VLSI Design, networking.

Manisha Y. Joshi completed Bachelor of Electronics Engineering in 1994 from BAMU Aurangabad, and Masters of Electronics Engineering in 1999 from SRTMU nanded, Ph.D. in 2015 (from SRTMU Nanded). She is member of IEEE, Fellow of Institution of Engineers (India), Fellow of IETE (India) and a life member of ISTE.her subjects of interest include Applied cryptography, Image Processing, Computer Algorithms, computer networks.

Lalitkumar Wadhwa completed Bachelor of Electronics & telecommunication Engineering in 1997 from Amaravati universiry, Masters of Digital Systems(E&TC) in 2000 from COEP,Pune University, and Ph.D. in 2016 (from IIT Dhanbad). He is member of IETE, Fellow of IETE (India) and a life member of ISTE. He is Principal of NMIET Pune since 2018. His subjects of interest include wired and Wirless Network, Voice Network and Communication.

