Secure Group Communication Using Binomial Trees

R. Aparna #1, B.B. Amberker *2, Divya Pola *3,Pranjal Bathia *4

Dept. of Computer Science and Engg. Siddaganga Institute of Technology, Tumkur, Karnataka, India.
[1] raparna@sit.ac.in

* *Dept. of Computer Science and Engg. National Institute of Technology, Warangal, Andhra Pradesh, India.*
[2] bba@nitw.ac.in, [3] divya.pola@gmail.com, [4] bathia.pranjal@gmail.com

***Abstract*—Numerous emerging applications like teleconferencing, scientific discussion etc. are relied on secure group communication model. Scalable group rekeying is one of the important issues in secure group communication since the group is dynamic in nature. Updating the group key and delivering it to group members securely must be carried out in an efficient manner, i.e., the number of encryptions performed and rekey messages constructed must be less. In this paper we propose a new method to manage secure group using binomial key tree approach. We show that number of encryptions performed and rekey messages constructed during membership change are less compared to the scheme proposed by Wong and others. In our scheme, it is not required to balance the tree after membership change. We compute the average encryption and rekey costs and show that the scheme is scalable.**

Keywords: Secure Group Communication, Binomial Tree, Encryption Cost, Rekey Messages, Key Server, Group Member.

I. Introduction

Secure Group Communication (SGC) deals with exchange of information between a set of users known as authorized users confidentially. Many emerging applications like scientific discussion, multimedia conferencing etc., are based on group communication model, in which members involved in such applications can send messages to others over an open network so that only authorized group members can have access to the message. Hence, communication among group members must be carried out securely.

Since group is dynamic in nature, key management is one of the important issue. A scalable secure group communication model ensures that whenever there is a membership change, new group key is computed and distributed to the group members with minimal computation and communication cost. The process of changing the group key and distributing it to group members is called *rekeying* and is carried out to provide *backward access control* during join and *forward access control* during leave. Thus, distributing the group key to members of the group either initially or during rekeying is an important issue in SGC. The computation cost, number of encryptions performed and the communication cost involved must be minimum and it improves scalability if it is independent of the group size.

Several group key management schemes have been proposed to support scalable SGC [1], [3], [4], [6]. Out of several group key management schemes, the scheme proposed by Wong et al. in [6] and Wallner et al. in [5] is most widely used. It uses a hierarchical key-tree approach (LKH) to manage a secure group and is considered as one of the efficient method which reduces the rekeying cost. In this approach if the key-tree is balanced, then it achieves logarithmic rekeying costs with respect to group size. However, since the group is dynamic in nature, members join/leave the group, the key-tree may become out of balance and remains unbalanced until some actions are taken to re-balance the tree.

In this paper we consider a binomial tree approach for managing the secure group. Unlike key tree approach which contains two types of nodes, binomial tree contains only one type of nodes called user nodes storing both user information and keys. In Binomial tree all subtrees are not always rooted at a single node. Instead, we get multiple subtrees and the number of subtrees entirely depends on the number of users in the group. In this approach, the tree never becomes unbalanced due to membership change. Hence, re-balancing techniques are not required.

II. Binomial Tree

A Binomial tree is defined as follows [2]: A binomial tree S_h is an ordered tree defined recursively. The binomial tree S_0 consists of a single node. The binomial tree S_h consists of two binomial trees S_{h-1} that are linked together: the root of one is the leftmost child of the root of other.

Binomial Key Tree: A binomial key tree is a binomial tree with users as the nodes. Binomial subtrees of height h have exactly 2^h nodes and there are exactly $\binom{h}{i}$ nodes at depth i, $i = 0, 1, , \ldots, h$. We consider a group of N users, u_1, u_2, ..., u_N sharing a common key known as group key. If the value of N is a power of 2, i.e., $N = 2^h$, where h is a non-negative integer, we get a single binomial tree, else we get multiple binomial subtrees with the size of each being a power of 2. We consider that all binomial trees are arranged in decreasing order of their height. Each node in binomial tree represents one user node. We name the subtrees as S_0, S_1, ..., S_{n-1}, where n is a non-negative integer, S_{n-1} being the tree with maximum height i.e., $n-1$ and S_0 with minimum height i.e., 0. A binomial tree S_h consists of 2^{h-i} number of subtrees S_i, $i = 0, 1, \ldots, h$. Therefore, there are a total of $2^{h+1} - 1$ subtrees. Users of each subtree form a subgroup B_j, $j = 1, 2, \ldots, 2^{j+1} - 1$. Further, users of subgroup B_j share a common key. Each user u stores the keys of all subgroups for which u is a member. If the users u_i, u_{i+1}, ..., u_j are in a single subgroup, B_g, $g = 1, \ldots, 2^{j+1} - 1$, of subtree S_t, a common key K_{ij} is shared among the users u_i through u_j.

Given a binomial key tree T, it specifies the secure group (G, K, R) as follows:

1) G denotes a finite and non-empty set of users, K denotes a finite and non-empty set of keys held by the users, and R is a binary relation between G and K, i.e., $R \subset GXK$, called user-key relation of the secure group. User u has the key k iff (u, k) is in R
2) T contains one type of nodes called user nodes storing both user information and keys

3) There is one-to-one correspondence between G and set of nodes in T
4) There is many-to-one correspondence between K and set of nodes in T
5) a) **T contains a single binomial tree, S_h:**
$(u, k) \in R$, iff u is a member of B_j, whose subgroup key is k, where $j = 1, 2, \ldots, 2^{h+1} - 1$. Note that the key of the subgroup with 2^h users itself is the group key.
b) **T contains m binomial subtrees:**
i) A group key $\overline{k} \in K$, $(u, \overline{k}) \in R$, $\forall u \in G$
ii) A subgroup key $k \in R - \{\overline{k}\}$, $(u, k) \in R$, iff u is a member of subgroup B_j of subtree i, where $j = 1, 2, \ldots, 2^{i+1} - 1$ and $i = 0, 1, \ldots, m - 1$.

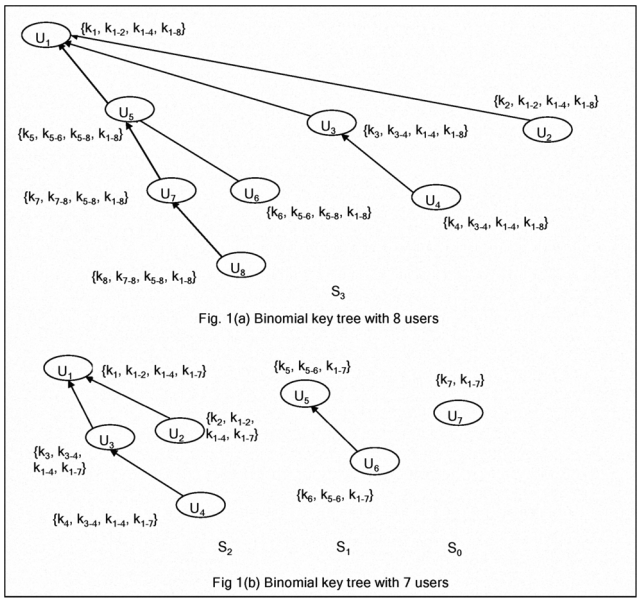

Fig. 1(a) shows a binomial key tree with 8 users u_1, u_2, ..., u_8 along with the keys held by each user. Since $N = 8 = 2^3$, we get a single subtree S_3 with the height being 3. We get $2^{3+1} - 1$ = 15 subgroups, B_1 through B_{15}. The subgroups B_1 through B_8 each contain one member u_1 through u_8 respectively, the subgroups B_9 through B_{12} have 2 members each, B_{13} and B_{14} contain 4 members each and B_{15} has 8 members. The relation $R = \{(u_1, k_1), (u_1, k_{1-2}), (u_1, k_{1-4}), (u_1, k_{1-8}), (u_2, k_2), (u_2, k_{1-2}), (u_2, k_{1-4}), (u_2, k_{1-8}), \ldots, (u_8, k_{1-8})\}$. Fig. 1(b) shows a binomial key tree with 7 users which contains 3 subtrees, S_0, S_1 and S_2. The subtree S_2 contains 7 subgroups B_1 through B_7, S_1 has 3 subgroups and S_0 has only one subgroup.

We consider a trusted key server responsible for group access control and key management. In particular, key server generates and distributes keys to group members and maintains the user-key relation.

Storage at each User: If the number of users, N, in the secure group is a power of 2, i.e., $N = 2^h$, then each user stores $h + 1$ keys. If not, each user stores $v + 2$ keys, $v = 0, 1, \ldots, h - 1$.

Storage at the Key Server: If $N = 2^h$, then storage at key server is $2N - 1$. If not, key server stores $2N - n - 1$ keys, where n denotes number of subtrees in the binomial tree.

III. Member Join Event

If a new user u_{new} wants to join the group, it sends a join request. Key server checks the number of users in the last subtree. If there is a single user, it inserts u_{new} into this subtree. If not, a new subtree with height 0 is created with u_{new} as the member. After inserting this new user, the number of subtrees may either get reduced or increased or even remain same. After reconstructing the tree (subtrees), all the keys of those subgroups whose membership is changed, must be changed to provide backward access control. After generating new keys, key server needs to securely distribute them to the group members.

Algorithm JOIN:
Let K be the old group key and K' be the new group key.
1) $u_{new} \rightarrow KS$: Join Request
2) $KS \leftrightarrow u_{new}$: Authenticate u_{new} and Distribute private key $K_{u_{new}}$
3)
- if $N/2 = 0$
  - u_{new} forms a new subtree with height 0
  - KS randomly generates 2 new keys
- else if $N = 2^w - 1$ (w is a non-negative integer)
  - one binomial subtree with height $h = w$ is obtained
  - KS randomly generates $h + 1$ new keys
- else
  - Attach u_{new} to the last subtree and reconstruct the tree if necessary
  - KS randomly generates $h + 2$ new keys

4) KS Distributes new group key K' by encrypting with previous group key and for user u_{new} with its private key
5) KS Distributes new subgroup keys to respective subgroup members by encrypting with appropriate keys.

Fig. 2 Algorithm for Join Protocol

Algorithm in Fig.2 presents the join protocol based upon this rekeying strategy. In the algorithm following assumptions are made: N: Number of users in the group before join operation, n: Number of binomial subtrees before join operation, m: number of binomial subtrees after join operation, h : Height of the last subtree after join operation. Performance of join operation can be measured in terms of number of new keys generated, number of distinct encryptions performed to convey new keys securely and number of rekey messages constructed. We encounter two different cases depending on the value of N : (i) N even and (ii) N Odd. In case of N odd, we encounter two subcases: $m = 1$ and $m \neq 1$. The complexity involved to insert a new user into a binomial key tree with N users is $O(logN)$

Advantages of Binomial key tree: One very good advantage in using binomial tree to manage SGC is it results in just 2 encryptions when the size of the group becomes odd after a new user joins the group. In general, if we consider a range of users, say, from user u_{2^h} to user $u_{2^{h+1}}$ join the group one after the other in sequence, then for half the number of joins, it requires just 2 encryptions. This is the best case for join operation. If the total number of users that join the group in sequence is $2^h = P$, then $P/2$ times it requires just 2 encryptions, $P/4$ times it requires just 4 encryptions, and so on. In general, $P/2^i$ times it requires $2i$ encryptions, where $i = 2, 3, \ldots, h$. When P^{th} user (i.e., user $u_{2^{h+1}}$) joins the group, it requires $2(h+1)$ encryptions.

IV. Member Leave Event

A user may leave the group either voluntarily or key server may expel the user from the group. In case of voluntary leave, the leaving member u_l sends a leave request to key server. Key server deletes the node u_l from the tree and reconstructs the binomial tree. The parent of leaving user u_l is called the *leaving point*. The number of subtrees may change after leave operation. If u_l is leaving from subtree i, $i = 0, \ldots, n-1$, key server generates the keys and distributes the new group key and subgroup keys securely to respective users to provide

TABLE I
COST OF ENCRYPTION AND REKEY MESSAGES FOR JOIN AND LEAVE OPERATIONS

	Binomial Key Tree scheme					Wong et al. scheme	
	Join operation			Leave operation		Join	Leave
	N Even	N Odd		$m = 1$	$m \neq 1$		
		$m = 1$	$m \neq 1$				
Cost of Encryption	2	$2h$	$2(h+1)$	$2h$	$2h + n - h_l - 1$	$2(h-1)$	$d(h-1)$
No. of rekey messages	2	$h+1$	$h+2$	$n+h-1$	$n+h-1$	h	$(d-1)(h-1)$
No. of new keys	2	$h+1$	$h+2$	h	$h - h_l + 1$	h	h

Algorithm LEAVE:

1) $u_l \rightarrow KS : \{Leave\ Request\}_{K_{u_l}}$
2) $KS \rightarrow u_l : \{Leave\ granted\}_{K_{u_l}}$
3) KS :
   - Find the binomial subtree S_h from which user is leaving, $h = 0, 1, \ldots, \lfloor log_2 N \rfloor$
   - For $i = h, h-1, \ldots, 1$, do
     If $u_l \in S_i$, Split the binomial subtree S_i into S_{i-1} and S'_{i-1}
   - Remove u_l from the binomial tree
   - Perform union of S_{h_l} with the subtree with height S_{h_l} which is obtained from split operation
   - for $i = 1, 2, \ldots, h-1$, do
     Perform union of two subtrees S_i to form S_{i+1}
   - If $m = 1$
     - then randomly generate h new keys
     - else
       randomly generate $h - h_l + 1$ new keys
4) Distribute new group key K' to roots of all binomial subtrees (i.e., roots of subtrees)
5) Distribute new subgroup keys to respective subgroups by encrypting with appropriate keys.

Fig. 3 Algorithm for Leave Protocol

forward access control. Algorithm in Fig. 3 presents the protocol for member leave event.

Following assumptions have been made in the algorithm: N: Number of users in the group before leave operation, n: Number of binomial subtrees before leave operation, m: number of binomial subtrees after leave operation, h: Height of the binomial subtree from which user u_l leaves, h_l: Height of the last binomial subtree before user u_l leaves. To measure the performance of member leave operation, we consider two different cases depending on the value of m: (i) $m = 1$ and (ii) $m \neq 1$. The complexity involved in removing a node from a binomial key tree with N users is also $O(logN)$

V. COST OF ENCRYPTION AND REKEY MESSAGES

Table I depicts the encryption cost, number of rekey messages constructed and number of new keys generated by the key server in order to convey changed keys to members of the group during membership change. In the table, N denotes number of users before join/leave, m is the number of subtrees after join/leave, h denotes height of the subtree in which join/leave occurs in case of binomial key tree and height of the tree in Wong et al. scheme and d denotes degree of the tree. Table II shows the average encryption and rekey message cost for a set of N users. Fig. 4 shows comparative graphs for encryption and rekeying costs in LKH and binomial key tree schemes. First two graphs of Fig. 4 show the encryption and rekeying costs during join operation. Graphs are plotted after considering an initial group size as 16 and computed the values after allowing users u_{17} to u_{32} to join the group one after the other. Similarly, the next two graphs of Fig. 4 depict the encryption and rekeying costs during member leave event for the same range of users. It is evident from the graphs that binomial key tree approach is efficient as compared to the Wong et al. scheme in terms of both encryption cost and number of rekey messages required.

TABLE II
AVERAGE ENCRYPTION AND REKEY MESSAGE COST FOR JOIN AND LEAVE

	No. of Rekey Messages	**Cost of Encryption**
Join operation	$3 + \frac{2}{N^2} + \frac{log_2 N - 2}{N}$	$4 + \frac{1}{2N^2} + \frac{4log_2 N - 1}{2N}$
Leave Operation	$1.25log_2 N$	$1.5log_2 N$

Fig.4 Comparative Graphs depicting encryption and rekeying costs for join and leave

VI. CONCLUSION

We proposed a new binomial key tree approach for managing the keys in Secure Group Communication. Binomial key tree contains only one set of nodes called user nodes storing both user information and keys. Storage required at each user is either less than or equal to that of LKH scheme. No balancing techniques are required after membership change. We showed that the number of encryptions required and rekey messages constructed in case of both join and leave operations are less compared to LKH scheme. Hence we conclude that the scheme is efficient and scalable.

REFERENCES

[1] Y. Amir, Y. Kim, C. Nita-Rotaru, J. Schultz, J. Stanton, and G. Tsudik. Secure Group Communication using Robust Contributory Key Agreement. *IEEE Transactions on Parallel and Distributed Systems*, 15,5, 2004, pp. 468-480.

[2] T.H.Cormen, C.E.Leiserson, R.L.Rivest, and C.Stein, Introduction to Algorithms, *PHI Publication, Second Edition*, 2006.

[3] D.A.McGrew and A.T.Sherman. Key Establishment in Large Dynamic Groups using One-Way Function Trees. *IEEE Transactions on Software Engineering*. Volume 29, Issue 5, pp. 444-458, May 2003.

[4] S.Rafaeli and D.Hutchison, A Survey of Key Management for Secure Group Communication, *ACM Computing Surveys*, 35(3), pp. 309-329, Sept. 2003.

[5] D.Wallner, E.Harder, and R.Agee, Key Management for Multicast: Issues and Architectures, *IETF, RFC 2627*, June 1999.

[6] C.K.Wong, M. Gouda, and S.S. Lam. Secure Group Communication Using key Graphs. *IEEE/ACM Transactions on Networking*, Volume 8, No.1, pp.16-30, Feb.2000.