International Journal of Computer Theory and Engineering, Vol. 2, No. 4, August, 2010
1793-8201

Efficient Distribution of Conference Key for Dynamic Groups

D. V. Naga Raju[2], Dr. V. Valli Kumari[1] and Dr. K. V.S.V.N. Raju[1]

Abstract—**The tremendous growth in the emerging technologies especially the increase in the bandwidth attracting the researchers for the development of new network based applications. One of the popular applications among them is the group based applications. The group based applications like e-learning, conferencing, TV over Internet, interactive gaming etc gaining popularity and influencing the life of modern people. These group based are vulnerable to many attacks. Maintaining security in multicast applications is a challenging issue. The dynamic changes in the group membership with frequent joins and evictions are the prime factor which makes the security difficult in any group based applications. Key distribution and key management is the challenging issue with any of these applications. This paper describes two novel techniques for secure distribution of the group key. The techniques proposed in this paper makes use of the hybrid key trees which allow the complete elimination of the secure channels for the distribution of the key material unlike many of the earlier proposed schemes, minimum storage requirements at each member, elimination of the chances of generation of week keys, less number of rounds and minimum computational overhead.**

Index Terms*—*Hybrid-keytrees, multicast, secure channel, computational overhead.

I. Introduction

In the information era, life became more easy and comfortable with extensive applications of Internet. The increased bandwidth and advancements in the technology are the forcing factors for popular growth of the Internet. It resulted in the development of attractive group based applications like stock quote updates conferencing etc. In group oriented applications data is to be sent to a legitimate group of receivers. Earlier people used to depend on point-to-point mechanism for delivering the data which is considered to be an inefficient mechanism with lot of network resource consumption. Though the network resources can be best utilized with multicasting, security concerns in multicast are more complex when compared to uni-casting with the number of increased participants. Group management is a critical issue in any group oriented application, as the frequency of joining and leaving of the members are very high.

Only legitimate multicast group shall be able to decrypt the data to maintain confidentiality in group communications. The confidentiality can be achieved by sharing a common key by all the members in a multicast group.

The development of group based applications should also satisfy various constraints like forward secrecy, backward secrecy, minimum bandwidth requirement, minimum storage requirement, and collusion freedom etc. Backward secrecy ensures security such that a new member is unable to access past communication. Similarly, forward secrecy ensures left members of the group also not have access to the future communication. Collusion freedom means that any set of evicted members are unable to deduce the current group key. In order to achieve the confidentiality, the group key should be changed with change in the member ship, which is called as rekeying. The efficiency of rekeying is an important issue in secure multicast as this is the most frequently performed activity with dynamic change in the membership.

The major issue in secure group communication is group key management. The existing solutions are classified as: centralized group key management, decentralized group key management, and contributory group key agreement. In centralized group key management protocols, one single entity key, known as Key Distribution Centre (KDC) controls the entire group. Key generation and distribution are carried out by a single unit. Each user holds two keys, Traffic Encryption Key (TEK) and Key Encryption Key (KEK). TEK is shared among the group members and is used for data transmission. KEK is shared between each user and the KDC and is used for the secure distribution of the key material. The TEK and KEK are to be refreshed with the membership change to maintain the forward and backward secrecy. Simplicity and efficiency are the advantages of these protocols. The weakness of scheme is the dependence on a single entity, which can be a single point of failure. The entire group will be affected if KDC is compromised. In decentralized group key management entire group is divided into subgroups thereby minimizing the problem of single point of failure. In this approach, hierarchies of key managers participate in the secure distribution of the group key. Distributed/contributory group key agreement does not place key generation burden on any one node. It takes the equal share of each user in the generation and distribution of the group key.

The rest of this paper is organized as follows. In Section-II we discuss related work. In Section -III we present the notations and terminology we used in this paper. In section-IV, we present our first scheme with its join, leave protocols. In section- V, we present our second scheme with its join, leave protocols. We analyze our protocols in section VI. We conclude in Section -VII.

[2]D. V. Naga Raju is with Shri Vishnu Engineering College for Women, Bhimavaram A.P, India (email:nagudatla@gmail.com).
[1]Dr. V. Valli Kumari is with Andhra University, AU College of Engineering, Visakhapatnam, A.P, India (email:vallikumari@gmail.com).
[1]Dr. K. V.S.V.N. Raju is with Andhra University, AU College of Engineering, Visakhapatnam, A.P, India (email:kvsvn.raju@gmail.com).

II. Related Work

Harney and Muckenhirn [4, 5] propose a group key management protocol, which uses the concept of pair wise keys. Each valid user shares a key (KEK) with the central server. The central server generates a Group Key Packet (GKP) that contains two keys: a Group TEK (GTEK) and a Group KEK (GKEK). For traffic encryption, GTEK is used. For the secure distribution of the GKP, GKEK is used. With a new member addition, the central server generates a new GKP and encrypts with the joining member's KEK and to the entire group encrypted with the old GTEK. When a member leaves the group, the key server generates a new GKP and sends it to each remaining member encrypted with the individual KEK that it shares with each member. This requires O (n) rekey messages.

Chinese Remainder Theorem based technique was proposed by Chiou and Chen [4]. The advantage of this scheme is that less number of rekeying messages is required. The disadvantage comes with the high computational overheads on the central server. C. K. Wong, M. Gouda, and S. S. Lam [1] proposed Local Key Hierarchy (LKH). Here a single entity known as KDC maintains a tree of keys. Each node is associated with a KEK. The leaves of the tree correspond to members of the group. The members are associated with the keys of the nodes in the path from its leaf node to the root. The key corresponding to the root of the tree is the TEK and is used for encrypting the message for group communication. Whenever there is a membership change, all the keys along the path from that member to the root need to be changed. The group key should also be changed. For a balanced binary tree, each member stores at most 1+log2 n keys, where n is the number of group members. The number of re-key messages is reduced in [1] when compared to [4], [5]. McGrew and Sherman [7] proposed an improvement over LKH, called One-way Function Trees (OFT). OFT allows reducing the number of re-key messages from 2log2 n to only log2 n. DeCleene et al. [6] proposed Intra-domain Group Key Management Protocol (IGKMP) which suffers from a single point of failure. The idea proposed by Inoue and kuroda [3] known as Fully Distributed Logical public key hierarchy (FDLKH), used the concept of LKH without any central server. In FDLKH, the members will not have any individual keys. Dynamically selected members known as captains take the burden of generation the distribution of the keys. The captains use DH key agreement for the generation of the key. Rafeli and Hutchison [8] proposed Hydra protocol. In this scheme for each sub group a controller (Hydra server) controls that subgroup. If a member joins a group or leaves from a particular group, the hydra server who is responsible for that subgroup in which the event has happened, takes the responsibility of generating and distributing to the other HS involved in that session. Rakesh Bobba, Himamshu Khurana [2] proposed DLPKH (Distributed Logical Public Key Hierarchy). DLPKH also used the concept of Logical Key hierarchy. DLPKH used public key trees for the secure distribution of the key material without establishing secure channels. Each member who occupies leaf node will hold all the private and public keys of their ancestor nodes and also the public keys of the nodes that are siblings to the set of nodes on the path from the leaf to the root. Dynamically selected Sponsors and co-sponsors are responsible for the generation and distribution of keys. The main objectionable issue with this scheme is that the private keys of ancestor nodes will be revealed to all the members. The scheme proposed in this paper does not reveal the private keys to all the members but only a trusted party.

III. Basic Principles and Assumptions

Assumes a binary tree. A logical key tree will be maintained like LKH [1]. Members occupy at leaf nodes. All the intermediate nodes possess the public keys of their ancestor nodes along with their key pair and group key. The members who occupy at leaf nodes know all the public keys of their ancestor nodes except the root's public key. The Members are divided into subgroups rooted at each intermediate node.

Notations

(l, m)	m^{th} node at level l in a tree
$(l, m)^I$	updated m^{th} node at level l in a tree1
M (l, m)	Member who occupies the node (l, m)
PK (l, m)	Public key associated with the node (l, m)
SK (l, m)	Private Key associated with the node (l, m)
PK^I	New Public Key associated with the node (l, m)
SK^I	New Private Key associated with the node (l, m)
T (l, m)	Sub tree rooted at the node (l, m)
E (PK, X)	Encryption of data X using a public key PK
E (SK, X)	Encryption of data X using a private key SK
GK	Old Group key
GK^I	New Group key
n	Number of members in a group
p,g	EIGmal group parameters
DH	Diffie-Hellman key exchange algorithm

IV. Protocols

A. Join protocol

Join protocol starts its execution when a member wants to join the group. The member who wants to join the group broadcasts the join request. This broadcast message consists of the public key of the joining member. The existing group members will decide the sponsor and the joining point. Sponsor will be the right most shallowest leaf node of the tree. There are two cases for joining: 1) if the joining point is a leaf node then two nodes will be created and these will be the children of the joining point. The sponsor associated with the joining point will be given to the right child (from root) and the public key of the new member will be given to the other node. 2) If the joining point is a non leaf node and has one child, this child is selected as sponsor. A new node is created and makes it the other child of the joining point and is given the public key of the new member. Diffie-Hellman key exchange algorithm is used by the sponsor and the root to exchange the new group key. The sponsor encrypts the new group key with the old group key and broadcasts. The sponsor also encrypts the new group key and all the updated public keys of the ancestor nodes of the newly joined member with the public key of the new member and sends it to the new member. The sponsor also

sends the complete tree structure to the new member.

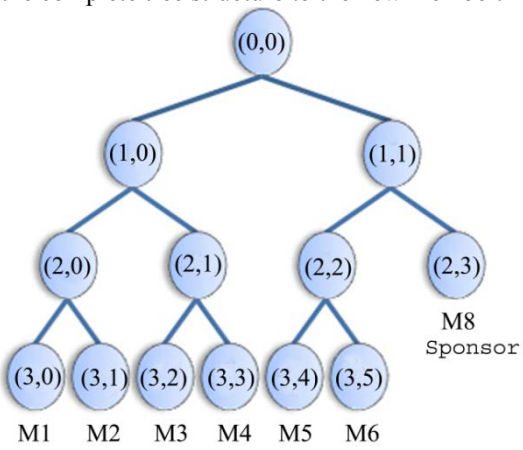

Figure 5: Tree before join

Sponsor (M8) →T (0, 0): E (GK, GK^I)
Sponsor(M8)→M7:E(PK of M7,(GK^I,$(2,3)^I$,$(1,1)^I$,$(0,0)^I$)

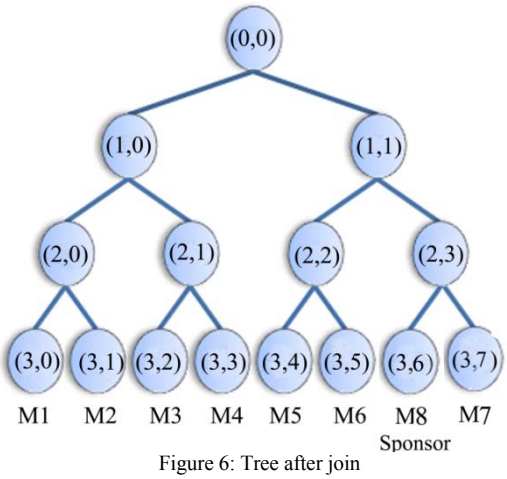

Figure 6: Tree after join

B. Leave protocol

When a member wants to leave, it sends a leave request to its parent. The parent broadcasts the leave request. This leave request includes the public keys of the ancestor nodes of the leaving member. Each node and member checks whether their public key or their parent's public key is there in the list or not. The node and a member whose parent's public key is available in the list but their public key is not there, send their private keys to the root (trusted third party) encrypted with the root's public key. Diffie-Hellman algorithm is switched between the root and the sponsor to exchange the new group key. Now the new group key will be broadcasted by the root encrypted with the private keys of the nodes who responded to the leave request. The root also encrypts the new group key with its own private key. The sponsors do not respond to the leave request. Now all the members and nodes get the new group key and they update the public keys of their ancestor's nodes.

New private key=old Private Key + GK^I mod p

New public key= old Public key $\times_g GK^I$ mod p

Sponsor selection: The sponsor is the right most shallowest (from root) leaf node of the tree. If the leaving member has a sibling then it is selected as the sponsor other wise the sponsor of the sub-tree rooted at the sibling node of leaving node's parent is selected as the sponsor.

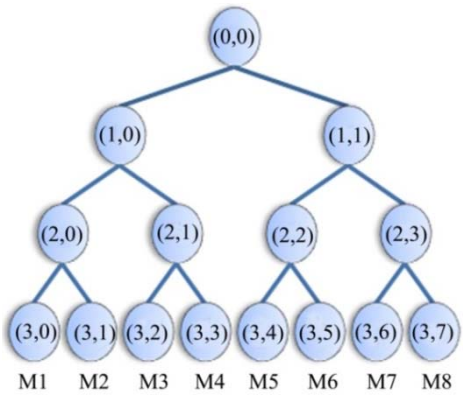

Figure 7: Tree before leave

Now suppose M8 wants to leave from the group, it sends a leave request to its parent (2, 3).The node (2, 3) broadcasts the leave request. The leave request consists of the public keys of its ancestors ((0, 0), (1, 1), (2, 3)). In the above figure 7, nodes(2,2) and (1,0) parent's public keys (1,1) and (0,0) respectively are there in the leave request but their public keys are not there so (2,2) and (1,0) respond and send their private keys to the root.

Here the sponsor (M7) and root (0, 0) will exchange the GKI using Diffie-Hellman algorithm.

Finally the root (0, 0) will encrypt the new GK with the private keys it has received during the leave request. The root also broadcasts the new GK encrypted with its own old private key.

Root →(1, 0): E (private key of (1, 0), GK^I)
Root →(2, 2): E (private key of (2, 2), GK^I)
Root →T (0, 0): E (old private key of root, GK^I)

Now all the members get new GK and also update their and their ancestor's public keys using the same formulae that were used during the join phase.

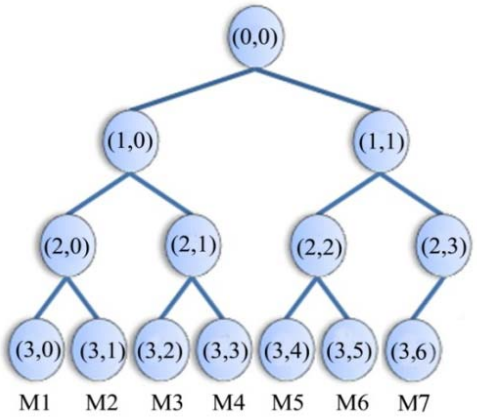

Figure 8: Tree after leave

V. Basic Principles And Assumptions (Scheme-2)

A logical key binary tree will be maintained like LKH [1]. Members occupy leaf nodes. Each member is associated with a key pair (public key and private key) and also a group key. Each member also maintains the public keys of all its ancestor nodes. The intermediate node holds its own private key and public key pair and a group key. The intermediate node also knows the public key of its immediate parent's sibling. Members are divided into subgroups rooted at each

intermediate node.

C. Join protocol

When a member wants to join the group, it broadcast a request with its public key. The current group members will decide the sponsor and the joining point. The joining point and the sponsor will be the right most shallowest leaf node of the tree. There are two cases for joining: 1) if the joining point is a leaf node then two nodes will be created and these will be the children of the joining point. The sponsor associated with the joining point will be given to the right child (from root) and the public key of the new member will be given to the other node. 2) If the joining point is not a leaf node and has one child then this child will be selected as sponsor and a new node will be created and make it the other child of the joining point and is given the public key of the new member. Diffie-Hellman key exchange algorithm is used by the sponsor and the root to exchange the new group key. The sponsor encrypts the new group key with the old group key and broadcasts. The sponsor also encrypt the new group key and all the updated public keys of the ancestor nodes of the newly joined member with the public key of the new member and sends it to the new member. The sponsor also sends the complete tree structure.

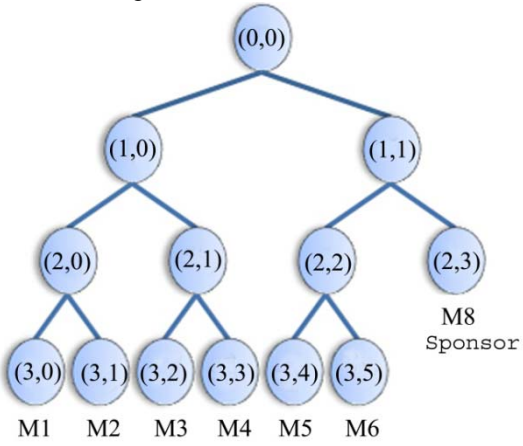

Figure 5: Tree before join

Sponsor (M8) →T (0, 0): E (GK, GK^I)

Sponsor(M8)→M7:E(PK of M7,(GK^I,$(2,3)^I$,$(1,1)^I$,$(0,0)^I$)

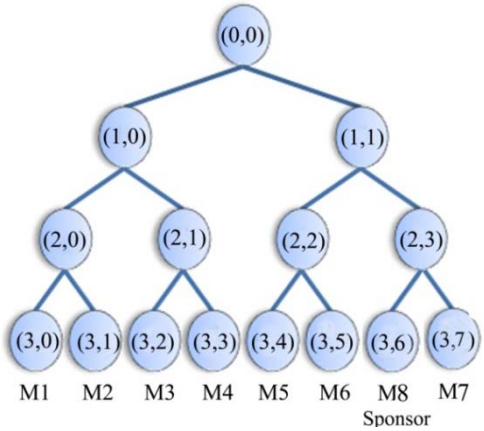

Figure 6: Tree after join

D. Leave protocol

When a member wants to leave, it broadcasts a leave request. The leave request includes the public keys of the ancestor nodes of the leaving member. Each node and member checks whether their public key or their parent's public key is there in the list or not. The node and a member whose parent's public key is available in the list but their public key is not there, will respond and send their private keys to the root(trusted third party) encrypting with the root's public key. Diffie-Hellman algorithm is switched between the root and the sponsor to exchange the new group key. Now the new group key will be broadcasted by the root encrypting with the private keys of the nodes who responded to the leave request. Unlike the previous scheme, the root need not encrypt the new group key with its own private key as the intermediate node knows the public keys of their parent's sibling. The sponsors will not respond for the leave request. Now all the members and nodes will get the new group key and they will update the public keys of their ancestor's nodes.

New private key=old Private Key + GK^I mod p

New public key= old Public key $\times_g GK^I$ mod p

Sponsor selection: The sponsor will be the right most shallowest (from root) leaf node of the tree. If the leaving member has a sibling then it will be selected as the sponsor other wise the sponsor of the subtree rooted at the sibling node of leaving node's parent will be selected as the sponsor.

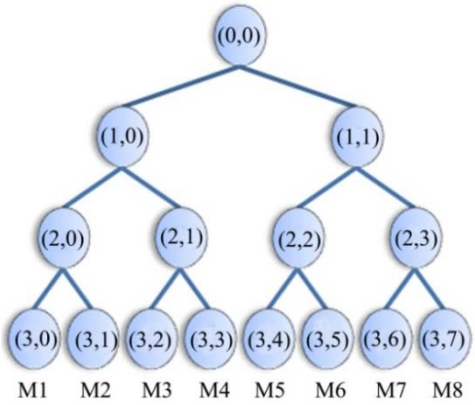

Figure 7: Tree before leave

Now suppose M8 wants to leave from the group, it sends a leave request to its parent (2, 3).The node (2, 3) broadcasts the leave request. The leave request consists of the public keys of its ancestors ((0, 0), (1, 1), (2, 3)). Each node and member checks whether their public key or their parent's public key is there in the list or not. The node and a member whose parent's public key is available in the list but their public key is not there will respond and send their private keys to the root encrypting with the root's public key. In the above fig. nodes(2,2) and(1,0) parent's public keys (1,1) and (0,0) respectively are there in the leave request but their public keys are not there so they(2,2) and (1,0)will respond and send their private keys to the root.

Here the sponsor is M7 and root (0, 0) will exchange the GKI using Diffie-Hellman algorithm.

Finally the root (0, 0) will encrypt the new GK with the private keys it has received during the leave request. The root also broadcasts the new GK encrypted with its own old private key.

Root →(1, 0): E (private key of (1, 0), GK^I)

Root →(2, 2): E (private key of (2, 2), GK^I)

Now all the members will get new GK and also update their and their ancestor's public keys using the same formulae that were used during the join phase.

Scheme	Rekeying	Data transfer
Proposed scheme-1&2	Both	Symmetric
DLPKH	Asymmetric	Asymmetric

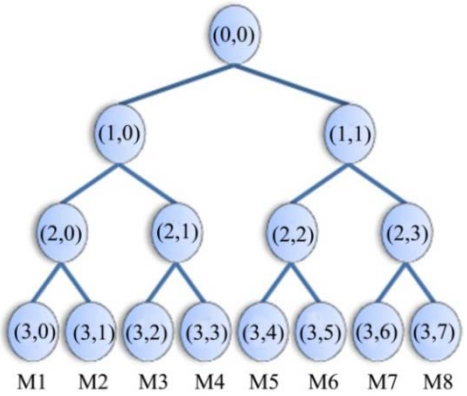

Figure 8: Tree after leave

VI. Analysis

A. Number of keys at each member

In the proposed scheme each members holds all the public keys of its ancestor nodes along with its own key pair and a group key where as in DLPKH each member holds all the ancestor nodes public keys, its own key pair and the public keys of its co-path nodes.

S.No	Protocol	Keys
1	Proposed Scheme-1	l+2
2	Proposed Scheme-2	l+3
3	DLPKH	3l+2

B. Total no. of rounds

The round complexity can be defined as the number of rounds it takes before termination.

S.No	Protocol	Join	Leave
1	Proposed Scheme-1	3	2
2	Proposed Scheme-2	3	2
3	DLPKH	3	2

C. Total no. of messages sent

S.No	Protocol	Join	Leave
1	Proposed Scheme-1	4	2l+2
2	Proposed Scheme-2	4	2l+1
3	DLPKH	2l+5	2l-2

D. Type of cryptosystem used for rekeying and message transfer

The proposed scheme uses asymmetric cryptosystem for sending the updated key material and symmetric cryptosystem in data transfer. DLPKH uses only asymmetric cryptosystem for both the operations. So computational complexity of the proposed system is less when compared to DLPKH.

VII. Conclusion

The critical issue in any multicast group communication is allowing access only to legitimate group members. LKH [1], FDLKH [3] requires the establishment of the secure channels for the distribution of the key material. DLPKH [2] eliminates the need of the secure channel establishment with the help of public key trees but suffers from the requirement of huge no of keys for each member as well as high computational resources. The ideas proposed in this paper also eliminates the need of secure channels for key distribution, the storage requirements for each member is also reduced and moreover less computational resources are required when compared to DLPKH. The advantage with our proposed scheme is that the private keys are not revealed as in earlier method [2] except to the trusted third party.

References

[1] C. K. Wong, M. Gouda, and S. S. Lam. Secure Group Communications Using Key Graphs. IEEE/ACM Transactions on Networking, 8(1):16–30, February 2000. DLPKH: Distributed Logical Public Key Hierarchy, Rakesh

[2] Bobba, Himamshu Khurana Volume 4812/2007 Springer Berlin /Heidelberg.

[3] FDLKH: fully decentralized key management scheme on a logical key hierarchy. Springer Berlin / Heidelberg, Volume 3089/2004

[4] G. H. Chiou and W. T. Chen. Secure Broadcast using Secure Lock. IEEE Transactions on Software Engineering, 15(8):929–934, August 1989.

[5] H. Harney and C. Muckenhim, Group Key Management Protocol (GKMP) Architecture," RCF 2094, July 1997.

[6] H. Harney and C. Muckenhim, Group Key Management Protocol (GKMP) Specification, RFC 2093, July 1997.

[7] McGrew and Sherman. Key Establishment in Large Dynamic Groups using One-way Function Trees, IEEE Transactions on Software Engineering- Volume 29 , Issue 5.

[8] S. Rafaeli and D. Hutchison. Hydra: a decentralized group key management. 11th IEEE International WETICE: Enterprise Security Workshop, June 2002.