A Secure Group Communication and Rekeying using Rabin's Squaring Trapdoor Function in Multicasting

D. Manivannan
School of Computing
SASTRA University
Tanjore, Tamil Nadu
India

A.R.Shloka
School of Computing
SASTRA University
Tanjore, Tamil Nadu
India

P.Neelamegam
School of Computing
SASTRA University
Tanjore, Tamil Nadu
India

ABSTRACT
In today's world of internet, secure communication among group of members has become vital, where multicasting plays an important role. In secure multicasting, security and scalability are the two important checks. Sometimes there is tradeoff between security and scalability. In order to transmit the data in a secure and scalable way, a suitable key management protocol should be implemented which reduces the number of rekey messages generated during the join or leave of any member thereby preserving forward and backward secrecy. In our proposed work, a new key management protocol has been proposed where a hierarchical structure has been implemented to improve the scalability. Derivation key is used to generate the new keys from the existing keys and Rabin's Squaring Trapdoor Function is used as the key derivation function to distribute the rekey messages. The security of the key derivation function lies in the hard mathematical problem of integer factorization which cannot be solved in a polynomial time. In order to increase the security, a salt value has been used along with the derivation keys for key stretching so that original keys used in the derivation function cannot be found out. The proposed protocol reduces the rekey messages by 1/d, in a d-degree tree and the number of modular exponential operations is only 2h compared to 3h in TGDH (Tree-based Group Diffie-Hellman) protocol, where h is the height of the tree.

General Terms
Key Management protocol for Group Communication.

Keywords
Rekeying, Key Derivation, Rabin's squaring Trapdoor Function, Pseudo Random Number Generator.

1. INTRODUCTION
With the widespread use of the internet and other web related activities in all the fields, the importance of group communication has grown into prominence. Members belonging to a group can remain in contact with each other and also one member can remain in contact with the other members of the group. This kind of communication is mostly used in some of the widely prevalent applications like teleconferences, distance education programmes and chat room applications. Once the member has moved away from the group it does not remain in contact with the other group members and the communication among the group members is kept as a secret from that non-member. These kinds of communication can be unicast, when only two members need to communicate, or broadcast or multicast, when a member wants to communicate with many other members. In multicast communication there is only single message packet transmission which reaches all the destined group members. The replication of the packets to be delivered to the group members is taken care by the nodes in the network. As the members are added to the group the messages are delivered to them also which helps in taking care of major issue namely scalability. Thus all multicast communications address the issue of scalability to some extent.

When a member joins the group it is given a multicast address ranging from 224.0.0.0 to 239.255.255.255. A member joining a multicast group can be communicated via UDP (Uniform Datagram Protocol) multicast only as the current technology permits. This makes it necessary to concentrate on two major factors namely reliability and security. Securing the data sent through multicasting network is very important because the multicast network is "open" in nature since the multicast address is public. So, anyone can access the multicast address and pose as a member and get hold of all secure data. The data sent through the multicast network are susceptible to attack by the intruders who may be within the group or outside the group. Hence there is needed to take care of the security issues namely Data confidentiality, Data integrity, and Data authenticity.

Simple cryptographic techniques of encrypting a message with a key and decrypting it does not suffice in today's environment in which the members in each multicast group is very large. Every time a member joins or leaves the group, the group key needs to be changed and communicated secretly to all the members thereby maintaining perfect forward and backward secrecy. There are several schemes namely Probabilistic schemes [11], Hierarchical schemes [3, 4, 5, 6], Conference Keying Schemes [10]. Of all the methods hierarchical methods offer a simple method to communicate data efficiently and secretly without much computations involved. While the probabilistic methods have huge storage overheads and the conference keying methods give unnecessary burden to the members who need to do huge exponential calculations. The hierarchical methods maintain a balance between both computation and storage overheads. We ask that authors follow some simple guidelines. In essence, we ask you to make your paper look exactly like this document. The easiest way to do this is simply to download the template, and replace the content with your own material.

1.1 Evaluation Criteria:
System Architecture must not be too complex that the implementation becomes difficult. The architecture considered is a binary tree which uses a key derivation function which is cryptographically secure and has a very simple implementation.

The number of rekey messages generated should be minimized. The hierarchical system is known to have minimum number of rekey messages whenever there is member addition or deletion. With the implementation of the key derivation function the rekeying cost is still reduced. Computational complexity should not be too high. The key derivation function used has an exponentiation of power 2 which is not very complex. The security of the system should be high. Rabin's Squaring function is known to be very secure since the security of the function lies in the mathematically hard problem of Integer factorization. Moreover finding the exact root of a number modulo n, where n is a very large composite number is infeasible.

This paper is organized as follows: In section 2 some of the existing protocols are analyzed, in section 3 some of the concepts that forms the background of the paper is discussed, in section 4 working of the proposed protocol is discussed, in section 5 performance of the proposed protocol is compared with the existing protocols, in section 6 the results are discussed.

2. RELATED WORKS

With the increase in need for multicasting with growing applications using internet, where data communication needs to be maintained secretly, there is always research in multicasting and key distribution. A simple approach includes Pair-wise keying protocols in which there exists a session key between each user pair using which they both communicate with each other. Though this protocol provides at most security, this does not scale well with large number of users since each user has to store n-1 session keys, when there are n members in the group. In Probabilistic methods [11] each user in the group is given random set of keys selected from the key pool. The users having the same key can communicate with each other. In case of member deletion, all the users sharing the key with the evicted member should change their key set. This protocol involves large storage overheads. In Hierarchical methods [3,4,5,6] a tree like structure is being followed in arranging the users. The root node is the group key which is obtained by all the group members. In Hierarchical Node based methods [1] the users of different efficiency are placed in different hierarchical levels. There is a group controller which designates the members of higher power to be sub-group controller (SGC). Each sub-group controller has a set of sub-group members who know the sub-group key (SGK) and communicate using this key. Intergroup communication is done by SGC by using Group key (GK). During member addition/deletion only the corresponding SGKs change and the rekeying cost is O (log m). In Hierarchical Key based methods [5, 6], the server maintains a tree structure and all the users are added in the root node. Each node is given a key called Key Encryption Keys (KEK) and the root node has GK. Each leaf node has a member with its individual key. Each member has all the keys from leaf to the root. In case of member addition/deletion all the keys on the path from root to the leaf has to be changed. If the degree of the tree is d then the rekeying cost is $d\log_d n$. The hierarchical protocol scales well when there is large number of users. In one-way function tree [3], the keys of both the child nodes are used for calculating the keys. For this both the nodes store the blinded node key of their siblings. When there is member addition/deletion all the keys on the path of that member is changed and cost is O (log n). But there is possibility of collusion attack which is tackled in [4]. In improved OFT [4]. All the blinded node keys on the path are also changed in case of member addition and deletion. This method avoids collusion attack but the rekeying cost becomes h^2. In the Tree Based Group Diffie-Hellman [10], which is the extension of 2-party Diffie-Hellman, all the group members calculate KEKs and the GK by themselves. This method can be efficiently used as conference keying protocol. Though there is no load on the server to distribute the keys, the members need to do large exponential calculation. It involves 2 rounds and 3 messages at the worst case during join/leave. Though the number of messages transmitted and the bandwidth used is minimum, it involves a total of 3h modular exponential calculations. In the shared key derivation method [12], one-way functions are used to find the new keys from the existing keys. Some of the members calculate the new key by themselves while for the remaining members the Server sends the updated keys. Compared to simple LKH this method reduces the rekeying cost.

3. BACKGROUND

The base structure of the proposed work is a simple binary tree. The server maintains the binary tree for storing the keys. All the members are added as the leaf nodes. The key in the root node is the group key. The keys in the leaf nodes is the individual key (IK) unique to each member. All the other keys are the Key Encryption Keys (KEK) which is used for sending the encrypted keys and not for sending the data. The data are communicated among the group by using the group key (GK). All the members have the keys which fall on the path from the leaf to the root. When a member is added or deleted all the other members sharing the keys with joining/ relieving member have to update their keys.

The proposed protocol uses concept of key derivation. Instead of the server generating the new keys, some of the old keys are used to generate the new keys. These keys are called the derivation keys. Any of the cryptographic function f (.), can be used as the key derivation function. The basic idea behind using the derivation keys and the key derivation functions is that some of the members can compute the new key by themselves and need not depend upon the server to give them those keys. Thus the computation and the transport overhead on the server have been reduced. The cryptographic function to be used as a key derivation function it should satisfy the following two lemma:

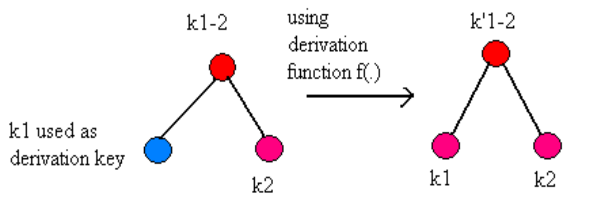

Figure 1: Key derivation

Lemma:

Given f(x) it is not computationally possible to compute x. This property is called the one-way property.

Lemma:

Given the different x1, x2….. which have been derived from the derivation function f(.), it is not possible to find f(.). This property is called the pseudo randomness property.

There are various cryptographic functions which could be used as key derivation functions namely the one-way hash functions, pseudorandom number generators and the one way trap door functions. Our protocol uses the Rabin's squaring one-way trapdoor function. The trapdoor one-way function is the one in which there exists trapdoor information, and only using that information, the function can be reversed.

Rabin's squaring one-way trapdoor function is used in the proposed protocol as pseudorandom number generator to update keys. The Rabin's function and its security can be explained as follows:

Let p, q be odd primes. Let n=pq. Then the squaring function SQUARE: $Z_p^* \rightarrow Z_p^*$ is given by,

$$\mathbf{f(n,x) = x^2 \bmod n}$$

The trapdoor information here is n=pq. This is a hard mathematical problem. The strength of the squaring trapdoor function lies in the fact that even when n is known , finding the factors of n is not possible and it is a hard mathematical problem and not possible to solve in polynomial time. When the congruence relation, $x^2 \equiv a \bmod n$ is taken, a is a quadratic residue. Using this congruence relation the solution for x can be a^{2m}, where m is any integer. Finding square root modulo n is as hard as factoring n. When the quadratic residuosity problem is hard, Squaring trapdoor one-way function can be used as perfect pseudorandom number generator.

The quadratic residuosity problem can be stated as follows: Let there be two very large distinct odd primes, p and q. Then n is an odd composite integer obtained as n=pq. Zn is a group having numbers from 1 to n-1, both numbers are inclusive. It can be said that aЄZn is a quadratic residue modulo n if and only if aЄQp, where a is a quadratic residue in the group 1 to p-1, and aЄQq, where a is quadratic residue in the group 1 to q-1. Thus Given a number 'a', finding whether it is a quadratic residue or a non-quadratic residue is possible only when the factors of n are known, which in turn is hard problem.

Even if it is known that a given number is quadratic residue, finding the square roots is difficult. There are four square roots for a given, $x^2 \equiv a \bmod n$ and finding the exact square roots is possible only when the factors are known. Alternatively when the four exact square roots are known the factors can be found in polynomial time. Thus finding the exact square roots of a square modulo n and the factors of n are computationally equal and both cannot be possible in a polynomial time when the other is unknown. Thus both are considered to be cryptographically hard.

Finally inside the key derivation function, a salt value k_g is used for key stretching along with the derivation keys to strengthen the key derivation function. This leaves the attacker to try every possible combination to find the original key thus increasing the security of the system further.

4. PROPOSED PROTOCOL:

4.1 Join/Leave Operations:

4.1.1 Initial Setup:

After a group of members offer their willingness to join the group, the server does the initial set up by authenticating all the members who wish to join the group and generates a binary tree. All the group members are added to the leaf nodes of the tree. The server then provides the members with the KEKs, GK and the IK.

4.1.2 Member Join:

When a member wants to join the group, the member sends a message to the server asking for its permission to join the group. The Server authenticates the member and gives it an individual key through a secure channel. The member is added to the rightmost shallowest node. If the tree is balanced then a new node is created and added. Then the server updates the entire key on the path from the leaf to the root of the added member. For key generation and updation the server uses derivation keys and key derivation function. The key derivation function used is the Rabin's Squaring trap-door function which can be used as the pseudorandom number generator. At each level (i) the server selects a key among the keys which is not to be changed, which act as the derivation key for the node at the previous level (i-1). This message is broadcast to all the users. All the users who have these keys calculate the new keys by themselves and the server has to send the messages only to the remaining members thus reducing the number of rekey messages.

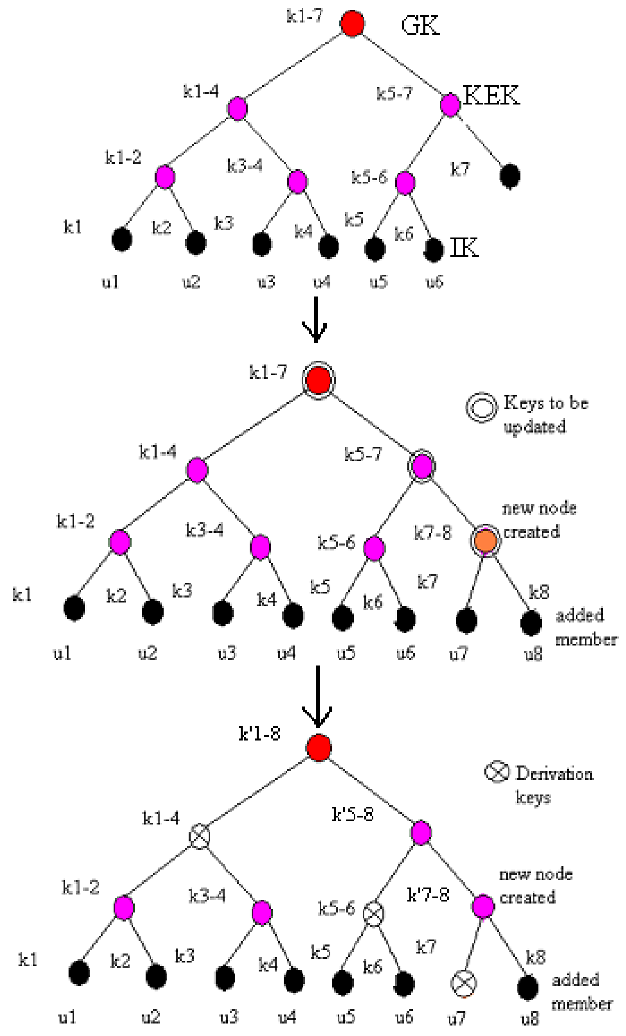

Figure 2: Key updation using key derivation

For example suppose user u8 joins the subgroup, rekeying takes place in the following steps.

Step 1: Selection of the derivation keys by the server from one of the keys which is not to be changed.

Step 2: Calculation of n= pq, Where p and q are odd primes and n is a large composite. It is mathematically hard to factorize the integers even after knowing n.

Step 3: The server generates the salt value, k_g, which is some random value in order to avoid repetitions in the keys that are generated.

Step 4: The server concatenates all these values and broadcasts the message as in Figure.3

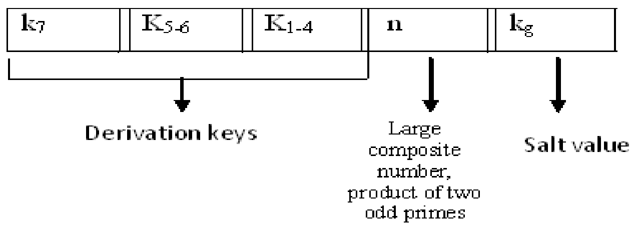

Figure 3: Broadcasted message for key derivation

Step 5: (in the member side)
The members look into their database and all the members having the derivation keys calculate the new keys themselves.
Now the user u7 can calculate

$K'_{7\text{-}8} = (k_7 \oplus k_g)^2 \bmod n$

u5 and u6 can calculate,

$K'_{5\text{-}8} = (k_{5\text{-}6} \oplus k_g)^2 \bmod n$

u1, u2, u3, u4 can calculate GK,

$K'_{1\text{-}8} = (k_{1\text{-}4} \oplus k_g)^2 \bmod n$

Step 6: (in the server side)
For the remaining users the server sends the appropriate keys as follows:
Server: $E_{k8}\{k'_{7\text{-}8}\} \rightarrow$ u8
Server: $E_{k'7\text{-}8}\{k'_{5\text{-}8}\} \rightarrow$ u7,u8
Server: $E_{k'5\text{-}8}\{k'_{1\text{-}8}\} \rightarrow$ u5,u6,u7,u8

4.1.3 Member Leave:

Whenever a member leaves the group it sends the message to the server. The server grants permission to leave and updates all the keys in the path of the leaving members. The procedure for key updation is same as for member addition.

5. COMPARISONS AND ANALYSIS

5.1 Transportation Overhead:

The method of key derivation substantially reduces the number of rekey messages. The number of rekey messages is given by log m, where m is the number of members in the group. For normal LKH2 it is 2logm (Figure.4, Figure.5). The rekey messages can be reduced to almost half when a key derivation function is used in a binary tree. This protocol can be extended to LKH of any degree d and it has been analyzed and found that almost there is reduction in 1/d rekey messages during join and leave (Figure. 6).

5.2 Computational Overhead:

The Squaring trapdoor function which has been used as key derivation function is an exponentiation function, but of power 2, hence feasible to compute. When the key derivation function need to be periodically changed the server can just compute the different n values by changing the p and q values instead of constructing a whole new function. Since strength of the squaring function lies on the hardness of the integer factorization, Rabin's squaring function can be used as a key derivation function. The number encryptions done by the server is also reduced since some of the members calculate the updated keys by themselves and the server need not send any encrypted messages to them.

The implementation of the Tree Based Group Diffie-Hellman protocol though drastically reduces the rekey messages it involves calculation of huge exponentiations on the member side which becomes unnecessary burden on the member side. The total number of modular exponentiations is also reduced to 2h, h is the height of the tree from 3h for TGDH (Figure.7). With the assumption that the server has greater power than the members it can take the burden of computing the new key and passing on to the section of the users as in our protocol.

5.3 Storage Overhead:

In the proposed protocol the number of keys stored in each member is same as LKH protocol i.e., all the keys from the member to the root node. But in the Tree Based Group Diffie-Hellman protocol the storage in each member is log m keys from the nodes in the path and also the blinded node keys of the siblings on the path, totally, 2log m. Thus the storage overhead is very high in this protocol compared to the proposed protocol.

6. RESULTS

The following table summarizes all the results in terms of storage and number of rekey messages during join and leave.

Table 1 Comparison of performances of various protocols

Criteria	**Simple LKH2**	**Tree based Group Diffie-Hellman**	**proposed protocol**
Total no. of keys in the server	2m-1	4m-3	2m-1
No. of keys in each member	$\log_2 m$	$2(\log_2 m)$	$\log_2 m$
Rekey messages during join	$2\log_2 m$	3	$\log_2 m$
Rekey messages during leave	$(2\log_2 m)-1$	3	$(\log_2 m)-1$

From the table it is seen that our protocol has less number of rekey messages without trading off on storage much. LKH has same storage but higher number of rekey messages. Tree-based

Group Diffie-Hellman protocol has only 3 rekey messages and 2 rounds but has a huge storage cost.

The number of modular exponential operations of our protocol are compared with Tree-based Group Diffie-Hellman protocol. It is found that our protocol has less modular exponential operations than TGDH and hence the computational complexity of our protocol is also balanced. The exponential operations are only of the power 2 and hence less computational overhead on the memberside whereas in TGDH members need to perform huge exponential operations.

Table 2 Comparison of computational complexity of proposed protocol and TGDH

Modular Exponential operations	Tree-Based Group Diffie-Hellman	proposed protocol
Join	3h (2h by the sponsor node and h by other nodes)	2h (h by members and h by server)
Leave	3h (2h by the sponsor node and h by other nodes)	2h (h by members and h by server)

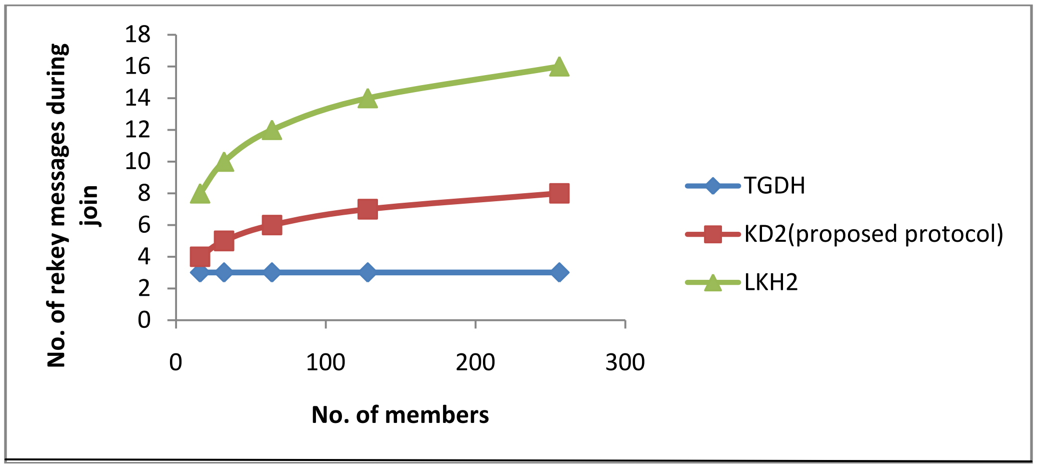

Figure. 4 Comparison of rekey messages during join

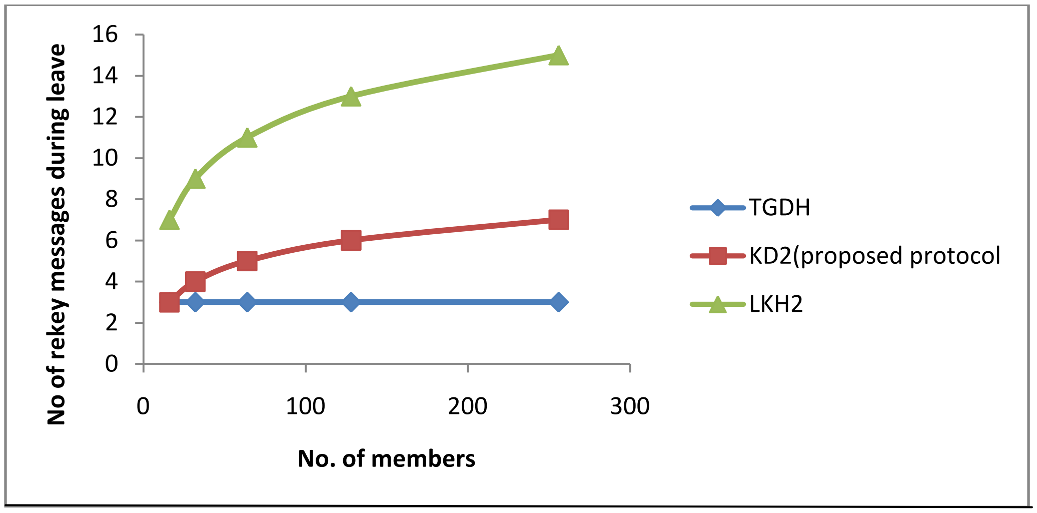

Figure. 5 Comparison of rekey messages during leave

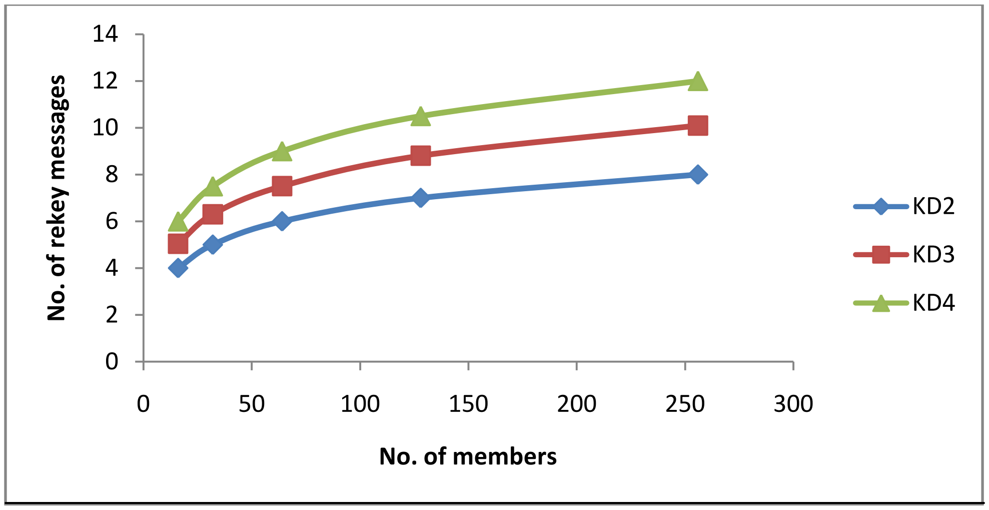

Figure. 6 comparison of proposed protocol for various tree degrees

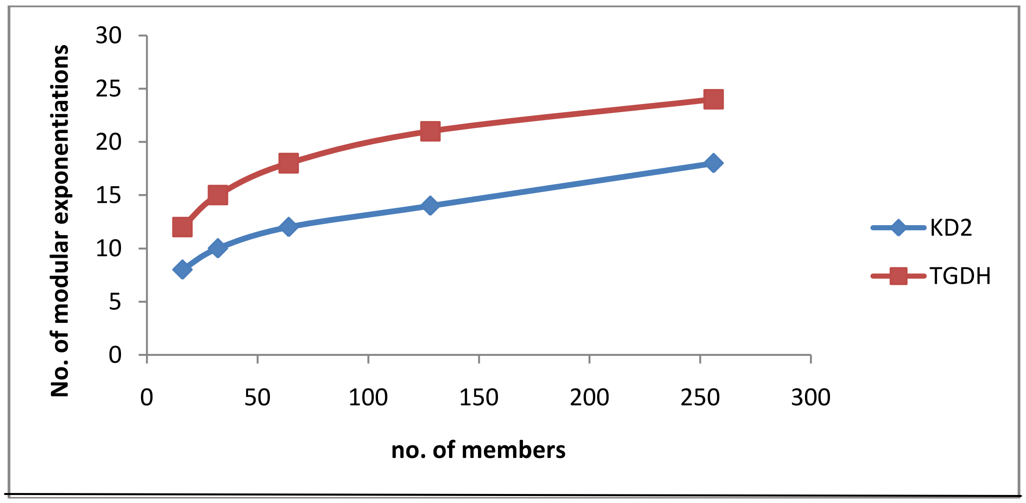

Figure.7 Comparison of number of modular exponentiations

7. CONCLUSION

In this paper an efficient group key management protocol has been proposed which is also scalable. The proposed protocol uses hierarchical framework which is known to have minimum number of rekey messages. Though the protocol can be extended to a tree of any degree, binary tree has been implemented which is known to be the simplest structure and very easy to implement. The proposed work uses the Rabin's squaring trapdoor function as the key derivation function and breaking this function is a hard mathematical problem. This function is thus used as a secure pseudorandom number generator . The salt value that is being used for key stretching along with the derivation key increases the randomness and also making the function more secure. The new key obtained is the quadratic residue modulo n, and it is very hard to find the roots of the quadratic residue modulo n. Even if there is a remote chance of finding the roots, the original derivation key is impossible to find with the use of the salt value in the squaring function. Thus the proposed protocol is secure and also scalable. The storage overhead is also balanced. The computation of the new key is also not very complex with only exponential modular calculations of power 2.

8. ACKNOWLEDGEMENT

Our thanks to the experts who have contributed towards the development of this research work. Our sincere thanks to our family and friends who supported and encouraged us.

9. REFERENCES

[1] S. Mittra, 1997. Iolus: "A framework for scalable secure multicasting," ACM SIGCOMM Computer Communication Review, vol. 27, no. 4, pp. 277-288.

[2] S. Rafaeli and D. Hutchinson, 2003. A survey of key management for secure group communication, ACM Computing Surveys, vol. 35, no.3, pp. 309-329.

[3] D. Balenson, D. McGrew, A. Sherman, Internet-draft, August 2000. Key management for large dynamic groups: one-way function trees and amortized initialization”, Internet Draft, IRTF.

[4] W.-C. Ku, S.-M. Chen, 2003. An improved key management scheme for large dynamic groups using one-way function trees, Proceedings of the IEEE International Conference on parallel Processing Workshops.

[5] H. Harney, E. Harder, August 1999. Logical key hierarchy protocol, Internet Draft, IETF.

[6] Wong, M. Gouda, S. Lam, 1998. Secure group communications using key graphs, Proceedings of the ACM SIGCOMM'98, ACM, pp. 68–79.

[7] Kin-Chin Chan, S.H.Gary Chan. Key management approaches to offer data confidentiality for secure multicast, Hong Kong University of science and technology.

[8] Mike Burmester, Yvo Desmedt. A Secure and Efficient Conference Key Distribution System.

[9] Matthew Moyer, Georgia Institute of Technology, Josyula R.Rao, Pankaj Rohatgi, IBM Thomas J.Watson Research center. A Survey of Security Issues in Multicast Communications.

[10] Yong dae Kim, Adrian Perrig, and Gene Tsudik. Tree-based Group Key Agreement.

[11] Laurent Eschenauer, Virgil D. Gligor. A Key Management Scheme for Distributed Sensor Networks.

[12] Jen-Chiun Lin, Kuo-Hsuan Huang, Feipei Lai, Hung-Chang Lee. Secure and efficient group key management with shared key derivation.

[13] July 1997. Group Key Management Protocol (GKMP) Architecture”, RFC 2094.