A Novel Scalable Group Key Management Protocol

Amrutasagar Kavarthapu
Gudlavalleru Engineering College
Gudlavalleru 521356
Andhra Pradesh, India

Seshagirirao Ganta
Gudlavalleru Engineering College
Gudlavalleru 521356
Andhra Pradesh, India

ABSTRACT
Secure and reliable group communication is an active area of research. The main issue in secure group communication is group dynamics and key management. Group key management brings challenges on scalability for multicast security. Member joining and member leaving from the group is the main challenge in designing secure and scalable group communication for dynamic update of keys. Most of the proposed solutions are not considering this parameter and so suffer from the one-affects-n scalability problem. In this paper, we present a new group key management protocol and express that it has better scalability when compared with other important protocols.

General Terms
Network security.

Keywords
Group communications, group key management, key, multicast security and scalability.

1. INTRODUCTION
Multicasting is nothing but delivery of messages to a group of destination computers simultaneously in a single transmission from a source. Now a day's Multicast applications have grown and greatly influenced our life along with the growth of the Internet. Examples of such applications include teleconference, information services, and distributed interactive simulation. As an important and mandatory building block for multicast applications, multicast security has been extensively researched in the past decades for protecting multicast communications. The research on multicast security addresses authentication, confidentiality, and access control, among other areas, where group key management is a key component. However, scalability is still a hard problem and a sizable challenge for group key management technologies. To make sure that group communication confidentiality, a group key management protocol must create and distribute a symmetric encryption key called traffic encryption key *(TEK)* or group key. In addition to the group key secrecy, the group key management protocol must provide forward secrecy and backward secrecy. Forward secrecy prevents an accessing current communication by old member after it leaves from the group. Backward secrecy prevents an accessing of the communication sent before a new member joins to the group. To do so, a rekey process should be performed after every join or leave operation within the secure group. It consists in generating a new *TEK* and distributing it to all group members. The main problem with any rekey technique is scalability: as the rekey process should be performed after every member join or leave, the computational and communication overhead induced may be important in case of frequent join and leave operation to group.

In this paper we propose A Novel Scalable Group Key Management Protocol based on Chinese Remainder Theorem (CRT) and a hierarchical graph B-Tree, in which each node consists of keys and a modulus. The new protocol reduces number of rekeying operations from log_2^n to log_m^n when compared with the SGKMP [1], where n is the number of leaf nodes and m is the order of the B-Tree. In this paper we are taken order of B-Tree is 3(i.e. m=3).

The remainder of the paper is organized as follows. Section 2 presents our new protocol. Section 3 presents compares the new protocol with others. Section 4 shows the testing performance of the new protocol. Section 5 gives our conclusion.

2. A NOVEL SCALABLE GROUP KEY MANAGEMENT PROTOCOL
Our novel scalable group key management protocol (NSGKMP) is based on the following: the Chinese Remainder Theorem and a hierarchical graph B-Tree, in which each node contains two keys and a modulus. The protocol is designed to minimize Re-keying operations.

2.1 Chinese Remainder Theorem
Let $m_1,m_2, ...m_n$ be n positive integers where they are pair wise relatively prime (i.e.$gcd(m_i,m_j)$=1 for i ≠ j, 1≤i,j≤n), $R_1,R_2, ...R_n$ be any positive integers, and $M=m_1m_2...m_n$. Then the set of linear congruous equations $X \equiv R_1 \ mod \ m_1, ...X \equiv R_n \ mod \ m_n$ have a unique solution as: $X= \sum_{i=1}^{n} R \ M_i y_i mod \ M$, where $M_i=M/m_i$ and $y_i=M_i^{-1} \ mod \ m_i$.

In the new protocol, the keys and moduli are constructed as a B-tree and maintained by the key server. Here each node of the B-Tree in the new protocol is assigned three values: two keys and a modulus. Figure 1 depicts the key and modulus graph, where *TEK* is a traffic encryption key,k_{ij} is a key encryption key, and m_{ij} is a modulus.

2.2 Moduli Maintenance
The key server needs to store $3log_3^n$ moduli and each member needs to store log_3^n moduli but they do not need to keep the moduli secret. The sibling nodes in the tree graph are assigned with three different moduli (i.e.m_{i1},m_{i2} and m_{i3} where i is the depth of the tree) and the nodes in the different level of the tree are assigned with the different moduli but each three of siblings at the same tree depth are assigned with the same three moduli under the different

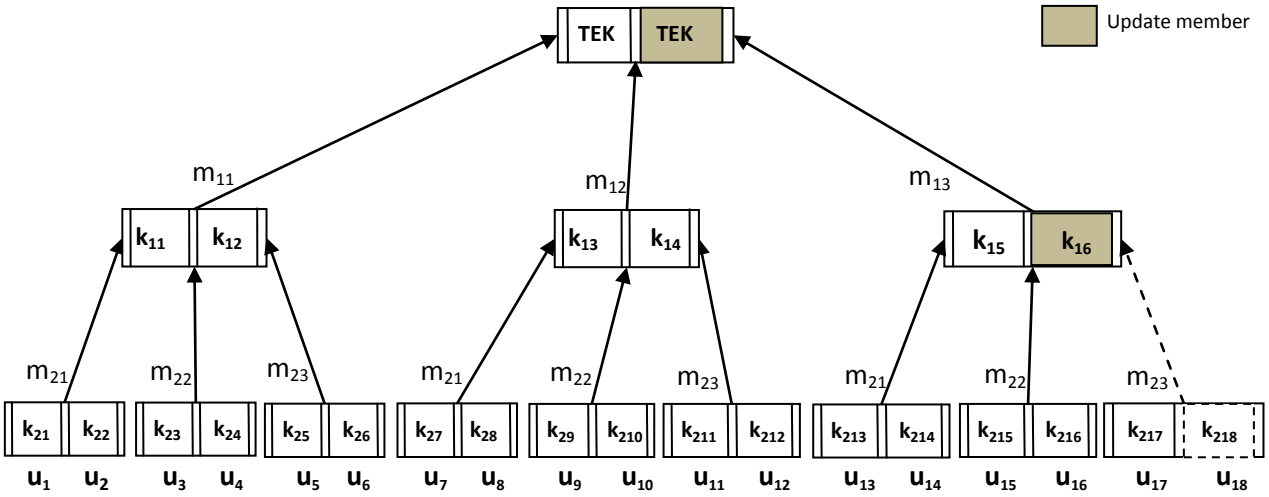

Fig 1: A B-Tree with nodes contains keys and modulus

parents (see Figure 1). This means there are only $3log_3^n$ different moduli in the B-tree graph, i.e.m_{ij} ($1 \le i \le log_3^n$, j=1,2,3) where i is the depth of the node in the tree, and the nodes (except the root) on a path from a leaf to the root and its direct children exactly cover all moduli. For instance, in Figure 1, for a path from u_1 to the root, the moduli on the path include m_{11} and m_{21}, and the moduli on its direct children include m_{12},m_{13},$m_{22}$and m_{23}. In addition, all different moduli in the tree graph should be pair wise relatively prime (i.e., gcd($m_{ij}\, m_{st}$,)=1 for i≠s or j≠t), and each modulus should be bigger than the key encryption value, i.e., $m_{ij} > E_{k_{il}}(k_{st})$ where m_{ij} and k_{il} belong to the same node and k_{st} belongs to its parent node.

2.3 Key Maintenance

The key server needs to store *3n-2* keys, i.e., TEK and k_{ij} ($1 \le i \le log_3^n$, $1 \le j \le 3^i$) where *i* is the depth of the node in the tree and *j* is the ordinal number of the node in the i^{th}depth of the tree, and each member needs to store log_3^n keys. The key server shares the keys with each member on the path from its leaf to the root. The keys on its path from the leaf to the root need to be updated in the protocol when a member joins or leaves the group but all moduli must be kept fixed[8].

To update the keys on the tree graph, the key server generates a new key for each update node and encrypts it with its children keys on its path from the leaf to the root. For instance, the key server needs to generate new keys $\{TEK', k'_{il}\}$ to update $\{TEK, k_{il}\}$ for the arrival of member u_d(its leaf key is k_{wd},$w=log_3^n$) to the group, where $1 \le j \le log_3^n$ and $l=\lceil (2i^2+1)d/n \rceil$, and encrypts the updated keys using the following formula,

Where

$e=\lceil (m^2-1)d/(n-i-1) \rceil$ and $v=\lfloor m^i d/n \rfloor$.

$$K_{st}=\begin{cases} E_{k_{st}}(k'_{il}) & \text{if } 1 \le i < log_3^n, t \ne e \\ & \text{where } s=i+1, t=3l-2 \text{ or } t=3l-1 \text{ or} \\ & t=3l \text{ or } t=3l+1 \text{ if } l \text{ is odd} \\ & \text{otherwise } t=3l-1 \text{ or } t=3l \\ E_{k'_{st}}(k'_{il}) & \text{if } 1 \le i < log_3^n, t=e \text{ where } s=i+1 \\ E_{k_{st}}(TEK') & \text{if } l \ne v \text{, where } s=1 \\ E_{k'_{st}}(TEK') & \text{if } l=v, \text{ where } s=1 \end{cases}$$

The key server then calculates a lock *L* as follows and multicasts the lock with the indices of keys (i.e., *st* in the following formula) to all valid members.

$L=\sum_{s=1}^{log_3^n} \sum_{t=z}^{z+5} K_{st}\, M_{sj}\, y_{sj} \ mod$ M.

Where

$z=6a(s-1)+1$

$$a=\begin{cases} \lfloor d/6 \rfloor - 1, & if\ d\ is\ 6\ or\ 12\ or\ 18\ or\ 24 \\ \lfloor d/6 \rfloor, & otherwise \end{cases}$$

$$j=\begin{cases} 1, & if\ t \equiv 1 mod\ 3 \\ 2, & if\ t \equiv 2 mod\ 3 \\ 3, & otherwise \end{cases}$$

$M=\prod_{s=1}^{log_3^n} \prod_{j=1}^{3} m_{sj}$, $M_{sj} = M/m_{sj}$ *and* $y_{sj} = M_{sj}^{-1} mod\ m_{sj}$.

Each member decrypts the updated traffic encryption key and related key encryption keys based on their own moduli and keys[8].

For the departure of member u_d from the group, the process is as same as the above except calculating K_{wd} (i.e., K_{wd}=0).

As an illustration, we are giving the following example for the Re-key process in Figure 1, where the member u_{18} requests to join the group. The key server generates new keys $\{TEK', k'_{16}\}$ to update $\{TEK, k_{16}\}$ and does the following encryption:

$K_{218} = E_{k_{218}}(k_{16})$, $K_{217} = E_{k_{217}}(k_{16})$, $K_{16} = E_{k'_{16}}(TEK')$, $K_{15} = E_{k'_{15}}(TEK')$.

The key server then calculates a lock as
$L=K_{218}M_{23}y_{23}+K_{217}M_{23}y_{23}+K_{216}M_{22}y_{22}+K_{215}M_{22}y_{22}+K_{214}M_{21}y_{21}+K_{213}M_{21}y_{21}+K_{16}M_{13}y_{13}+K_{15}M_{13}y_{13}+K_{14}M_{12}y_{12}+K_{13}M_{12}y_{12}+K_{12}M_{11}y_{11}+K_{11}M_{11}y_{11}\ mod$ M, where $M=m_{11}m_{12}m_{13}m_{21}m_{22}m_{23}$, $M_{ij}=M/m_{ij}$, $y_{ij} = M_{ij}^{-1}\ mod\ m_{ij}$.

In the protocol, we can see that the key server uses the same modulus (M) and parameters (M_{ij}, y_{ij}) to calculate the lock for any Re-key process but the key encryption value (i.e.,K_{st}) for calculating the lock are changed based on the Re-key requested by the different members. This means the key server can pre-calculate the modulus (M) and parameters (M_{ij}, y_{ij}) to be used for later Re-key processing steps and only needs to calculate them once for a fixed tree graph.

3. SCALABILITY OF GROUP KEY MANAGEMENT PROTOCOLS

In order to measure the scalability of group key management protocols more accurately, we propose the following scalability metrics: 'storage', Level of processing Difficulty and 'number of Re-keying operations'. Storage measures the total number of keys maintained by the key server. The Level of processing Difficulty indicates applicability for small mobile devices. The number of rekeying operations is the number of new keys are generated due to a member join or leaving from the group. Table 1 gives a comparison on level of processing difficulty. Table 2 gives a comparison of the new protocol with the SGKMP[8]. Table 3 gives sample values for log_m^n.

In [1] binary tree structure is used. When the group is large, the number of levels in the binary tree will be more which increases number of keys at member. Extending this scheme to B-Tree will reduce the height of the tree reducing number of keys at each member. At the same time we should consider server side storage i.e. number of keys at the level of the tree. In [1] log_2^n keys are maintained by the every member in the tree, extending the scheme to B-tree will result in maintaining log_m^n keys by the members of the B-tree (where m is the order of B-tree). In [3] m-way tree structure is used, it will also reduce the height of the tree, here m keys are maintained by the each member, but in B-Tree scheme the member need to maintain only log_m^n keys. In [3] number of keys at server in m-ary tree in terms of d can be represented as $m*(d/log_2^m)$, where d is the height of the tree. According to [3] m-ary tree can maintain less number of keys at server when $m \leq 4$, but in B-tree scheme if m increases it will maintain less number of keys in respect to number of members in the group.

From the Table 2 we see that NSGKMP reduces the number of Re-keying operations from log_2^n to log_3^n when compared with [1], NSGKMP can store more number of keys when compared with the [1].

From the Table IV we see that NSGKMP need to store less number of keys at the key server when compared with the [3].

Table 1. A comparison of Level of Processing Difficulty

Protocols	GKMP	Secure Lock	LKH	SGKMP	NSGKMP
Level of processing Difficulty	Low	High	Low	Low	Low

Table 2. A comparison of NSGKMP with SGKMP

Scalability metrics	Number of Re-keying operations		Storage
Protocols	J	L	
NSGKMP	log_3^n		3n-2
SGKMP	log_2^n		2n-1

J: Join; L: Leave

Table 3. Sample values for log_3^n

n / m	2	3	4	5	6	7	8
2	1	1.58	2	2.32	2.58	2.80	3
3	0.63	1	1.26	1.46	1.63	1.77	1.89
5	0.43	0.68	0.86	1	1.11	1.20	1.29

Table 4. Comparison of NSGKMP with NSMGKM

Number of keys need to maintain at the key server	NSGKMP	NSMGKM
m		
2	3.16	4
3	2	3.79
5	1.36	4.31
7	1.12	5

4. PERFORMANCE OF THE NEW PROTOCOL

In this session we provide overview of simulation model and some of the results by comparing NSGKMP with the protocol of [1].From Figure 2 NSGKMP has the less number of Re-keying operations when a member joins or leave from the group. From Figure 3 NSGKMP needs to store less number of keys when compared with the [3].

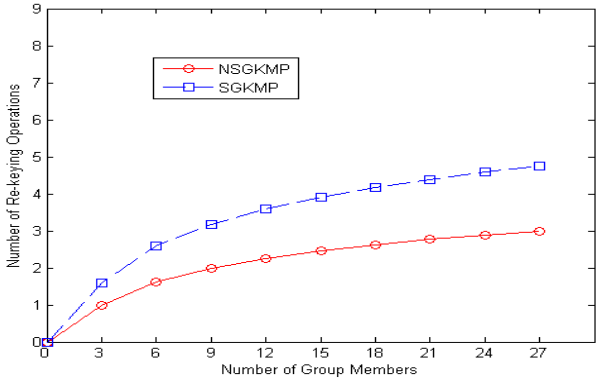

Fig 2: Number of Re-keying operations Vs group size

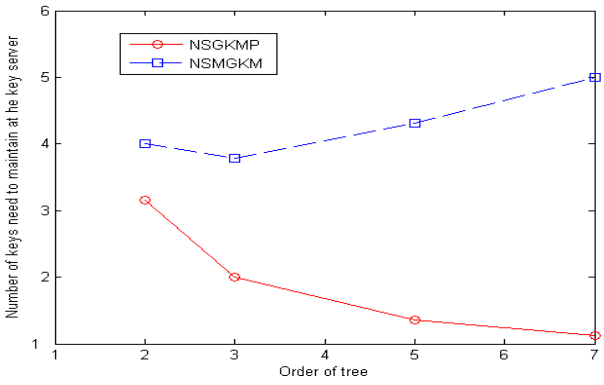

Fig 3: Order of tree Vs Number of keys needs to maintain at the key server

5. CONCLUSION

To improve the scalability of the Group key management we propose A Novel Scalability Group Key Management Protocol and demonstrates it has better scalability in terms of number of Re-keying operations and storage (from Calculations in Table II). If we increase the order of B-Tree then we can automatically decrease number of Re-keying operations further more.

6. REFERENCES

[1] Ronggong Song and George O. M. Yee, "A Scalable Group Key Management Protocol (SGKMP)", IEEE Communication Letters, VOL. 12, NO. 7, pp.541-543, JULY 2008.

[2] D. Wallner, E. Harder, and R. Agee, "Key management for multicast: issues and architecture," National Security Agency, RFC 2627, June 1999.

[3] R. Varalakshmi and Dr. V. Rhymend Uthariaraj, "A New Secure Multicast Group Key Management (NSMGKM)Using Gray Code" IEEE –International Conference on Recent Trends in Information Technology, ICRTIT 2011, Anna University, june 2011.

[4] G. H. Chiou and W. T. Chen, "Secure broadcast using secure lock," *IEEETrans. Software Engineering*, vol. 15, no. 8, pp. 929–934, Aug. 1989.

[5] H. Harney and C. Muckenhirn, "Group Key Management Protocol (GKMP) architecture," RFC 2093, July 1997.

[6] H. Harney and C. Muckenhirn, "Group Key Management Protocol (GKMP) specification," RFC 2094, July 1997.

[7] D.SAMANTHA, Classic Data Structures, Prentice-Hall of India Private Limited, New Delhi-110001, 2006.

[8] M. Rameeya and S. Oswalt, "Tree Based Scalable Secure Group Communication". Bonfring International Journal of Research in Communication Engineering, Vol. 1, Special Issue, December 2011.