I. J. Computer Network and Information Security, **2015, 8, 49-55**
Published Online July 2015 in MECS (http://www.mecs-press.org/)
DOI: 10.5815/ijcnis.2015.08.06

A Centralized Key Table based Communication Efficient Group Key Management Protocol

Manisha Y. Joshi
M.G.M.'s College of Engineering, Nanded (India), 431602
Email: manisha.y.joshi@gmail.com, joshi_my@mgmcen.ac.in

Rajankumar S. Bichkar
G.H. Raisoni College of Engineering and Management, Pune (India),
Email: bichkar@yahoo.com

Abstract—Group key management is an integral part of secure multicast. Minimizing the number of rekeying messages, maintaining the forward and backward secrecy has always been a challenging task. Though there are many solutions which reduce the rekeying messages from $\mathrm{O}(\mathrm{n})$ to $\mathrm{O}(\log_2 \mathrm{n})$, they increase with the increase in group size. In this paper, we present a centralized key table based communication efficient group key management protocol in which number of rekeying messages is independent of the group size. In this protocol key management server (KMS) divides a group of n members into n subgroups of size $\mathrm{n}-1$ and maintains a table of n subkeys along with member ID and one group key. Each member has $\mathrm{n}-1$ subkeys, which is a subset of n subkeys of KMS and one group key. The proposed protocol requires only one multicast rekeying message per joining of a new member as well as per eviction of any existing member. As the number of rekeying messages is not dependent on group size, it requires less computation.

Index Terms—Group key management, rekeying, subkeys, secure multicast, forward and backward secrecy**.**

I. INTRODUCTION

Many collaborative applications use IP multicast. IP multicast does not provide security against access by unauthorized group members to access the group communication. Group communication is encrypted by group key also called as traffic encryption key (TEK) to protect the content from unauthorized members. To distribute and update the group key becomes an issue, when group is dynamic, i.e. a new member joins and an existing member leaves group frequently. The member who leaves the group should not be able to access the group content or reveal the current group key known as forward secrecy. The member who joins will not be able to access previous group content or reveal previous group key, known as backward secrecy. To maintain the forward secrecy and backward secrecy, group key has to be changed after every new member joins and every existing member leaves. This process is called as rekeying. The minimization of rekeying messages and computation complexity are still big challenges in group key management. Logical key hierarchy based solutions [2, 3, 4, and 5], reduces these messages using key tree to $\mathrm{O}(\log_2 n)$, where n is size of group. Performance of these schemes is optimal if the key trees are balanced. Balancing of the key tree is an issue. In this paper we propose the communication efficient group key management protocol based on centralized key table, which reduces the rekeying messages to $\theta(1)$ from $\mathrm{O}(\log_2 n)$. In this protocol key management server (KMS) divides a group of n members into n subgroups of size $n-1$ and maintains a centralized flat table of n sub-keys along with member ID and one group key. Each member has $n-1$ sub keys, which is a subset of n subkeys of KMS and one group key. In this, after eviction of any member there is one subgroup key which, isolates the evicted member and rest of group. This key can be used to encrypt the new group key. Hence, one multicast message will be sufficient to convey the new key and the computations will be moderate at server side. Rest of the paper is organized as follows. Section II presents the analysis of previous related protocols. Section III describes the proposed protocol along with rekeying process after join and leave. Section IV presents the security analysis of proposed protocol. Performance analysis and comparison with other well-known schemes are presented in section V. Section VI presents the conclusions of the paper.

II. RELATED WORK

Group Key Management Protocol GKMP [1] provides simple solution. In which, when a new member joins, new key is encrypted by old group key and multicast to old members. This is efficient when new member joins but when any member leaves the group, KMS has to send n encrypted rekeying unicast messages. Logical key hierarchy (LKH) tree based protocol proposed by Wong et al.. [2] and Wallner et al. [3] reduce the rekeying messages to $\mathrm{O}(\log_2 n)$. In the logical key hierarchy approach, group of n user is divided into small groups of size equal to tree order. In a tree based structure total number of rekeying messages increase with the group size. Sharman et al. [4] proposed the one way function tree (OFT) for secure multicast, which reduces the rekeying cost by half of the LKH proposed by Wong et al.. Jing Liu et al.[5] proposed the collusion resistance one way function tree to improve the security of scheme by Sharman et al..

In both approaches (LKH and OFT) balancing a key tree is a major issue. Many researchers [6, 7, 8, 9, and 10] proposed the solution to balance the tree. To keep the tree balanced after a new member joins or existing member leaves, key encrypted keys (KEK) or internal key nodes are rearranged. This rearrangement causes extra rekeying messages required to convey the group members about their new keys. This causes to extra encryptions i.e. computations on server as well as group member side.

Wald Vogel et al. [11] have proposed a different approach, than hierarchical key tree that is of maintaining flat table by KMS. The flat table consists of one Traffic Encryption Key (TEK) and 2*w* Key Encryption Keys (KEK), where *w* represents the number of bits in the member *id*. There are two keys (KEK) assigned for each bit in the member *id*, one associated with each possible value of the bit (0 or 1). Each group member receives *w* keys, associated with the state of its bit of its member ID. A member knows only the KEKs associated with bits in its id. When existing member leaves all KEKs known by members are changed. Table 1 shows a Flat table for maximum group of size 16. Each member will have four bit *id*. For example member of *id* 1101 will have set of keys {KEK_{01}, KEK_{10}, KEK_{21}, and KEK_{31}}.

Table 1. Centralized flat key table for 4 bit id

	TEK	
ID bit #0	KEK_{00}	KEK_{01}
ID bit#1	KEK_{10}	KEK_{11}
ID bit#2	KEK_{20}	KEK_{21}
ID bit #3	KEK_{30}	KEK_{31}
	Bit value=0	Bit value=1

Total number of keys stored by KMS as well as group members in any centralized group key management protocol is dependent on how the group members are divided into subgroups. For every subgroup one key is assigned called as key encryption key (KEK) or subgroup keys.

Fig. 1 shows set representation of keys and sub groups in LKH for the instance of group size 4. Where, group members are U_1, U_2, $U_3, and\ U_4$. Equation (1) shows set of keys shared by individual group members as follows. Hence when U_1 leaves the group KMS has to change the keys $K_1, K_{12}\ and\ K_{14}$.

$$\left.\begin{aligned} U_1 &\to \{K_1, K_{12}, K_{14}\} \\ U_2 &\to \{K_2, K_{12}, K_{14}\} \\ U_3 &\to \{K_3, K_{34}, K_{14}\} \\ U_4 &\to \{K_4, K_{34}, K_{14}\} \end{aligned}\right\} \tag{1}$$

$$U_1 \cap U_2 \cap U_3 \cap u_4 = \{K_{14}\} \tag{2}$$

$$U_1 \cap U_2 = \{K_{12}, K_{14}\} \tag{3}$$

$$KMS \to \{K_1\ K_2\ K_3\ K_4, K_{12}, K_{34}, K_{14}\} |KMS| = 7 \tag{4}$$

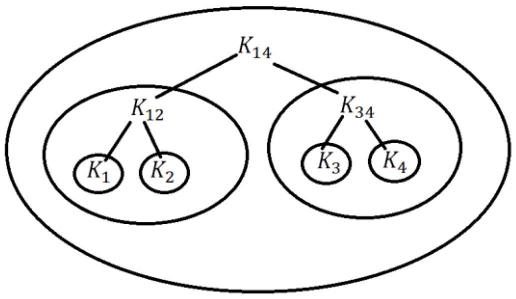


Fig.1. LKH set representation - Set of subgroup keys and group key, shared by each group member for group size 4.

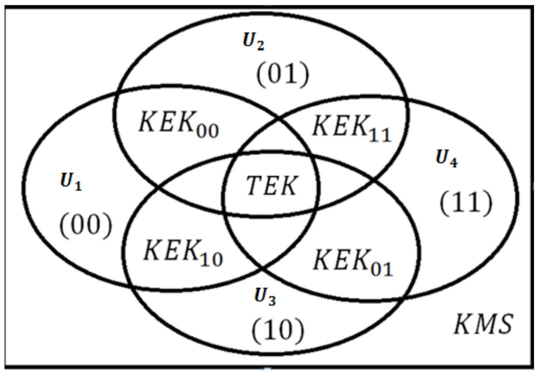


Fig.2. CFT set representation- Set of subgroup keys and group key, shared by each group member for group size 4.

Fig. 2 shows the set of subkeys called as *KEKs* and group key *TEK*, shared by individual group member $U_1, U_2, U_3, and\ U_4$. Where, *id* of these group member are of two bits, as group size is 4 as shown in table 2. Table 3 shows group members, their id and set of keys each of them stores. Total numbers of keys stored by KMS are 5, i.e.

$$KMS \to \{KEK_{00}, KEK_{01}, KEK_{10}, KEK_{11}\ and\ TEK\} \\ |KMS| = 5 \tag{5}$$

In the CFT scheme, number of keys stored by group members are 3 same as in LKH scheme. KMS stores fewer keys (i.e. 5) in CFT than LKH (i.e.7). Hence CFT is storage efficient. When user joins or leaves, KMS has to change the keys which are shared by that member. Number of rekeying messages is linearly proportional to the bits in group member id i.e. $\log_2 n + 1$. In group size 4 it is$(\log_2 4 + 1) = 3$, as shown in Table 3.

This scheme is storage efficient as number of keys stored by server are $2\log_2 n$ much less than LKH based approach but it is prone to the collusion attack, as two or more evicted members together can have total set of keys.

Table 2. Centralized flat key table for 2 bit id (group size 4)

	TEK	
ID bit 0	KEK_{00}	KEK_{01}
ID bit 1	KEK_{10}	KEK_{11}
	$Bit\ value\ = 0$	$Bit\ value\ = 1$

Table 3. No of keys shared/stored by each group member in group of size 4

Group Members	Id	Set of Keys
U_1	00	$\{KEK_{00}, KEK_{10}, TEK\}$
U_2	01	$\{KEK_{00}, KEK_{11}, TEK\}$
U_3	10	$\{KEK_{01}, KEK_{10}, TEK\}$
U_4	11	$\{KEK_{01}, KEK_{11}, TEK\}$

Chinese remainder theorem (CRT) based protocol was proposed by Chou et al. [12] and other protocols such as SKPCRT [13] reduces number of rekeying messages to $\theta(1)$, but at server side it has to do heavy computation to create a secure message. R. Song et al. [14] have proposed the Chinese remainder theorem and LKH together, to reduce the computation cost and increase the scalability. The scheme requires only one message for rekeying. But the storage and computation increases and also the bandwidth required to transmit the message is increased. Recently, Vijaykumar et al. [19] proposed the group key management system using CRT for batch rekeying, which reduces the computations at sever side.

To overcome this tradeoff between computational overhead, storage cost and rekeying messages [15, 16, and 17], we have proposed a new communication efficient group key management protocol. In this protocol, a new approach of making $n-1$ subgroups is proposed. In this, after eviction of any member there is one subgroup key which, isolates the evicted member and rest of group. This key can be used to encrypt the new group key. Hence, one multicast message will be sufficient to convey the new key and the computations will be moderate at server side. The proposed protocol is secure against the collusion attack.

III. Proposed Protocol

In the proposed protocol, group members and key management server (KMS) maintains a key table. Key table consists of a group key, called as traffic encryption key (TEK) and $n-1$ subgroup keys along with the ID of that group member as shown in Fig.3 and Fig.4.

Fig. 3 shows the set representation of proposed scheme for group size 4. Where, each group member stores three subkeys and one group key (TEK). Each group member has set of keys as follows

$$\begin{array}{c} U_1 \rightarrow \{SK_2, SK_3, SK_4 \text{ and } TEK\} \\ U_2 \rightarrow \{SK_1, SK_3, SK_4 \text{ and } TEK\} \\ U_3 \rightarrow \{SK_1, SK_2, SK_4 \text{ and } TEK\} \\ U_4 \rightarrow \{SK_1, SK_2, SK_3 \text{ and } TEK\} \\ KMS \rightarrow \{SK_1, SK_2, SK_3, SK_4 \text{ and } TEK\} \\ U_4 = KMS - \{SK_4\}\ |KMS| = 5 \end{array} \qquad (6)$$

When a group member U_1 is evicted other members have key SK_1 which U_1 does not have. A new group key can be encrypted using SK_1 and multicast to other members except U_1 .

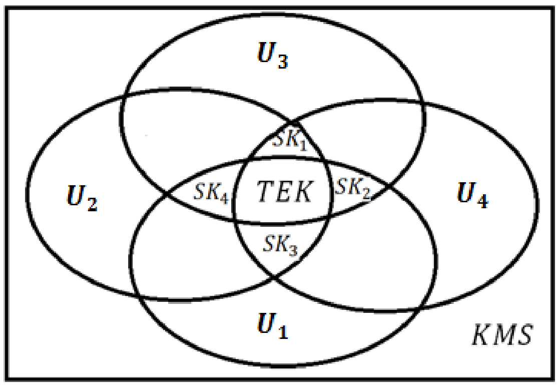


Fig.3. Set representation of proposed scheme- Subgroups and set of keys shared by each user and KMS for group size 4.

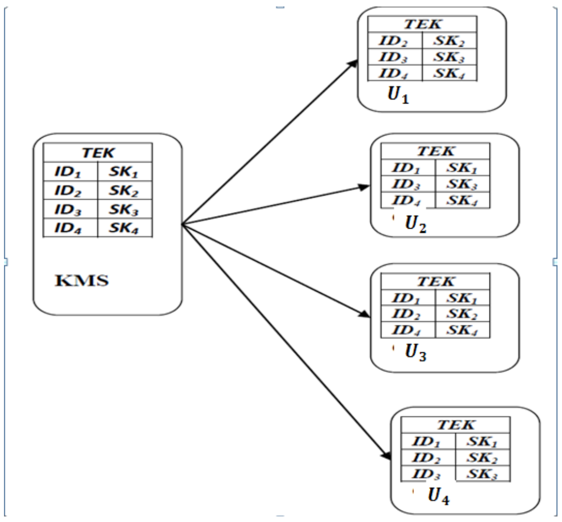


Fig.4. Initial group of size 4.each group member maintains a key table.

A. *Initial setup*

Each group member (GM) registers with key management server (KMS) and shares a common secret key using any authenticated protocol like SSL [18]. Initially for a group of n members KMS generates n subgroups of size $(n-1)$ and $n+1$ keys for n subgroups and one main group may be called as traffic encryption key (TEK). Let, U_i , be the i^{th} member of group G such that,

$$\begin{array}{c} G = \{U_1, U_2, .. U_n\}, \quad S_i = \{G\} - \{U_i\} \\ for\ 1 \le i \le n \end{array} \qquad (7)$$

Where, S_i is the i^{th} sub group $for\ 1 \le i \le n$

Each member U_i receives group key, TEK (traffic encryption key) and set of $n-1$ subgroup keys (SGK). Where SGK $\{SK_j \mid j \neq i, 1 \le j \le n\}$ and set of operations $\{+, \vee, \oplus, \ll_l\}$ here, $a \ll_l b$ means a is shifted left by l least significant bit value of b i.e.

$$101110111110 \ll_4 011011010010 = 11101111100$$

Fig.5 shows an instance for a group of four user i.e. $n = 4$.

B. Join

Fig. 6 shows the message exchange between KMS and group members, when a new member joins the group. The KMS and group members have to perform following steps:

1. A new member U_{n+1} sends a join request to the KMS
2. KMS establishes a common secret key K_{n+1} with new member using standard protocol like SSL[18]
3. KMS generates a new group key TEK' and multicast to all existing members by encrypting with old group key i.

$$[E_{TEK}\{TEK' \parallel r \parallel o_1 \parallel o_2\} \parallel join \parallel ID_{new}] \quad (8)$$

Where, r is randomly generated 128 bit integer, o_1 and o_2 are also randomly generated integers between the range 1 to 4 ($o_1 \neq o_2$) that denotes the operation to be performed in the next step, i.e.

$$o_1, o_2 = \{1,2,3,4\} = \{+, \vee, \oplus, \ll_l\} \quad (9)$$

4. KMS updates its key table of SGKs (SK_i for $1 \leq i \leq n$) using function

$$SK_i' = f(TEK', SK_i, r) = (SK_i o_1 TEK') o_2 r \quad (10)$$

5. KMS assigns old group key as new SGK and enters in key table. i.e. $SK_{n+1} = TEK$,as shown in Fig. 5(b) and Fig.6
6. Each existing member receives new group key TEK'. Each one updates its sub group keys (SGK) using equation (9), For the value of o_1 or $o_2 = 4$ i.e $\ll_l$ $l = r \bmod 8$
7. Each user adds old group key TEK as new SGK i.e.$SK_{n+1} = TEK$ in its key table.
8. KMS delivers the new group key TEK' and set of updated SGK $\{SK_j \mid 1 \leq j \leq n\}$ to the new user.

C. Leave.

Fig. 7 explains the message exchange between KMS and group members, when an existing member leaves the group. The KMS and group members have to perform the following steps.

1. If ith member leaves the group, it sends leave request or intimation.
2. KMS generates new group key TEK' .
3. As ith member does not have subgroup key SK_i but rest of the members have, KMS encrypt new group key TEK' using SK_i and send multicast message to rest of the members

$$[E_{SK_i}\{TEK' \parallel r \parallel o_1 \parallel o_2\} \parallel leave \parallel ID_i] \quad (11)$$

4. KMS deletes SK_i from its key table as shown in one instance of Fig. 5.
5. KMS updates its subgroup keys SK_j for $1 \leq j \leq n-1$ using function Similar to step 4 in join process

$$f(TEK', SK_j) = (SK_j o_1 TEK') o_2 r \quad (12)$$

6. All group members except ith member will be able to decrypt the message. Replace old group key with new group key TEK'
7. All group members except evicted member will update its sub group keys using new group key and received r and operator o_1 and o_2 using same function of KMS

$$f(TEK', SK_j) = (SK_j o_1 TEK') o_2 r \quad (13)$$

8. All members delete the subgroup key SK_i from their key table.

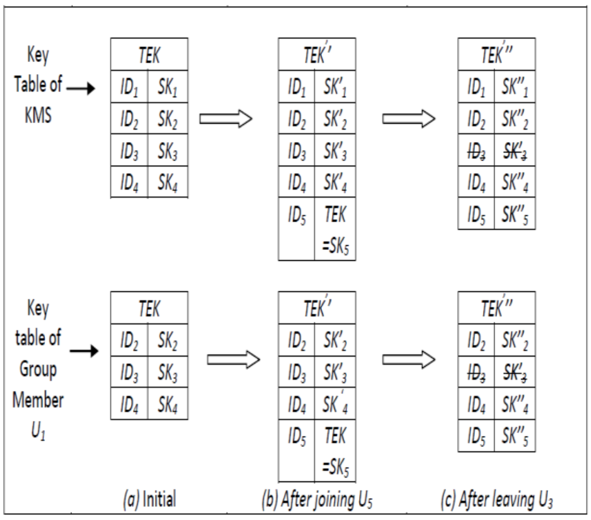
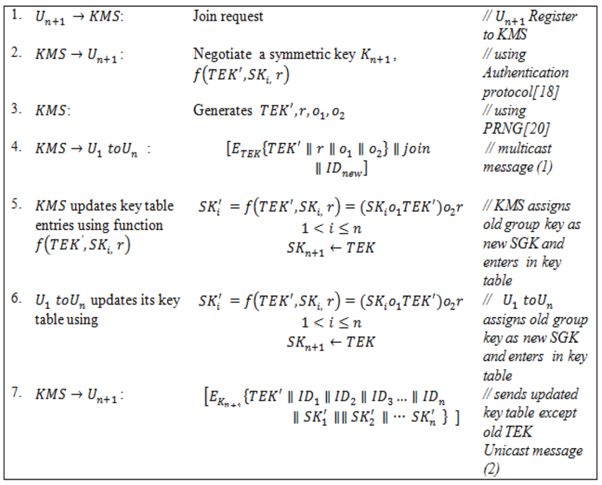
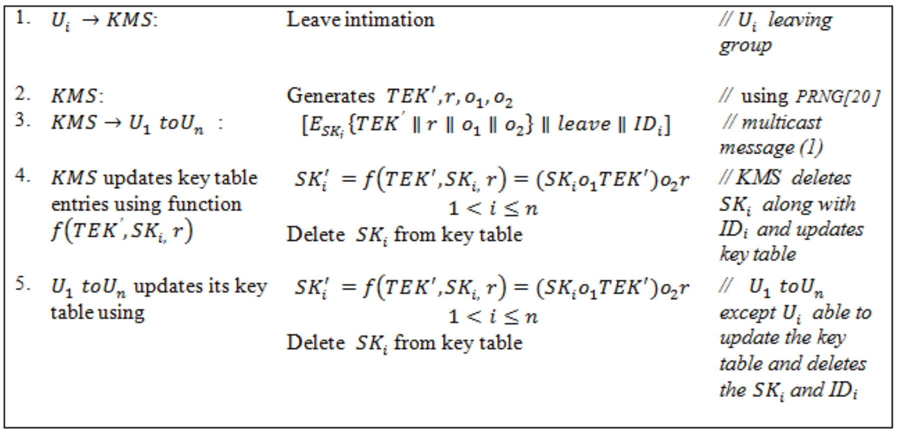


Fig.5. Key Table at KMS and Group Member U_1 for group size of four. (a) Initial table, (b) After joining new member U_5 , both KMS and u1 updates its key table. (c) Key table of after leaving group member U_3



1.	$U_{n+1} \rightarrow KMS$:	Join request	// U_{n+1} Register to KMS
2.	$KMS \rightarrow U_{n+1}$:	Negotiate a symmetric key K_{n+1}, $f(TEK', SK_i, r)$	// using Authentication protocol[18]
3.	KMS:	Generates TEK', r, o_1, o_2	// using PRNG[20]
4.	$KMS \rightarrow U_1$ to U_n :	$[E_{TEK}\{TEK' \parallel r \parallel o_1 \parallel o_2\} \parallel join \parallel ID_{new}]$	// multicast message (1)
5.	KMS updates key table entries using function $f(TEK', SK_i, r)$	$SK_i' = f(TEK', SK_i, r) = (SK_i o_1 TEK') o_2 r$ $1 < i \leq n$ $SK_{n+1} \leftarrow TEK$	// KMS assigns old group key as new SGK and enters in key table
6.	U_1 to U_n updates its key table using	$SK_i' = f(TEK', SK_i, r) = (SK_i o_1 TEK') o_2 r$ $1 < i \leq n$ $SK_{n+1} \leftarrow TEK$	// U_1 to U_n assigns old group key as new SGK and enters in key table
7.	$KMS \rightarrow U_{n+1}$:	$[E_{K_{n+1}}\{TEK' \parallel ID_1 \parallel ID_2 \parallel ID_3 \ldots \parallel ID_n \parallel SK_1' \parallel\parallel SK_2' \parallel \cdots SK_n'\}]$	// sends updated key table except old TEK Unicast message (2)

Fig.6. Message exchanges between KMS, group members and $\boldsymbol{U_{n+1}}$, when $\boldsymbol{U_{n+1}}$ joins



1.	$U_i \rightarrow KMS$:	Leave intimation	*// U_i leaving group*
2.	KMS:	Generates TEK', r, o_1, o_2	// using *PRNG[20]*
3.	$KMS \rightarrow U_1$ $to U_n$:	$[E_{SK_i}\{TEK' \parallel r \parallel o_1 \parallel o_2\} \parallel leave \parallel ID_i]$	*// multicast message (1)*
4.	KMS updates key table entries using function $f(TEK', SK_i, r)$	$SK_i' = f(TEK', SK_i, r) = (SK_i o_1 TEK') o_2 r$ $1 < i \leq n$ Delete SK_i from key table	*// KMS deletes SK_i along with ID_i and updates key table*
5.	U_1 $to U_n$ updates its key table using	$SK_i' = f(TEK', SK_i, r) = (SK_i o_1 TEK') o_2 r$ $1 < i \leq n$ Delete SK_i from key table	*// U_1 $to U_n$ except U_i able to update the key table and deletes the SK_i and ID_i*

Fig.7. Message exchanges between KMS and group members, when U_i leaves the group.

IV. Security Analysys

A. *Backward secrecy*

In this scheme, all sub-keys are totally independent of each other. When a new member joins, all updated sub-keys are delivered to the new member. The new group key is not derived from old key. A new member does not have access to the old group key and thus cannot access previous content. Hence, backward secrecy is maintained.

B. *Forward secrecy*

When member U_3 from scenario shown in Fig. 4 and Fig.5, leaves the group, a multicast message for future key is encrypted using sub-key SK_3 and released by KMS i. e.

$$[E_{SK_3}\{TEK' \parallel r \parallel o_1 \parallel o_2\} \parallel leave \parallel ID_3] \qquad (15)$$

As member u_3 is not aware about SK_3 and SK_3 is not derived from any known value, which U_3 has, U_3 cannot decrypt the new message i.e. new TEK', so the forward secrecy has been maintained.

C. *Collusion attack*

When a member u_3 or any single member leaves, it has a set of sub-keys $SK_1, SK_2, and\ SK_4$. these three sub-keys are updated using following function by server and rest of the members.

$$SK'_j = (SK_j \ll_l TEK') \oplus r \qquad (14)$$

Suppose evicted member wants to retrieve new updated key SK_j'. It is having only SK_j. Knowing only variable/key SK_j it is not possible to retrieve SK_j'using any deterministic algorithm. Hence, it is resistant to collusion attack.

V. Performance Analysis Of The Proposed Protocol

In order to analyze the performance of our protocol, we have used three parameters,

1) Storage cost in terms of number of keys stored by key management server (KMS) and group members (GM),
2) Computation cost in terms of number of computations carried out by KMS and GM, and
3) Communication cost is measured in terms of number of rekeying messages.

Table 4 shows a comparison of performance of proposed protocol with other group key management protocols. It has been observed that proposed scheme has significantly less communication cost than Logical Key Hierarchy (LKH) and Centralized Flat Table (CFT). It has less computation cost than CRT based scheme proposed recently [13, 14]. Storage cost of KMS is less than other schemes on KMS side but it is greater at than other schemes at GM side. This may be considered as a limitation of our scheme.

Fig. 8, shows a graph of the numbers of rekeying messages per join and leave for efficient LKH protocol [10], Centralized Flat Table (CFT) of 10 bits ID and proposed protocol. Proposed protocol requires less number of rekeying messages compared to LKH and centralized Flat table. These results are simulated, over a scenario, where the initial group members are 1000, and equal number of joins/ leaves randomly over the range 100 to 600 from the group. The performance of proposed protocol is measured on Intel Pentium 4 CPU 4.00 GHz, 3 GB RAM for 1000 group members. It is observed that computation time at GM side for group size of 1000 is less than 10 msec.

Total numbers of computations are proportional to the total number of encryptions as encryption algorithms consume more computations compare to other mathematical operations. Hence we have measured total number of encryptions carried out by KMS per join and leave operations.

We have simulated the scenario of group size 10 to 2000. Total encryptions for a single join over the various group sizes as shown in Fig.9 and for single leave in Fig.10 are measured. Number of encryptions for efficient LKH [10] and centralized Flat Table of 16 bits are measured.

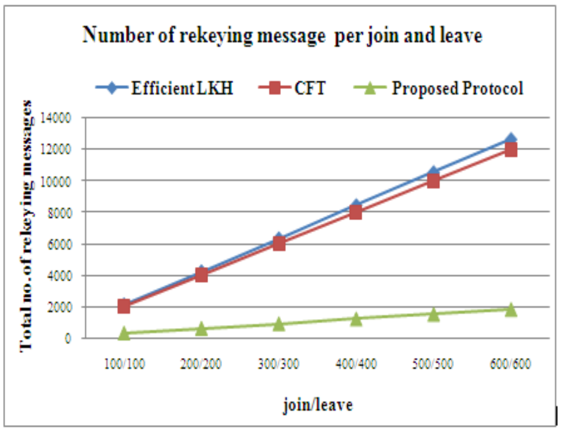


Fig.8. Number of rekeying messages per joins and leaves for LKH protocol, Centralized Flat Table of 10 bits ID and Proposed Protocol.

VI. CONCLUSIONS

In this paper we have proposed a communication efficient group key management protocol which uses centralized key table of subgroup keys along with traffic encryption key (TEK). In this protocol, the communication complexity is reduced to $\theta(1)$, as number of rekeying messages required per join and leave are constant i.e. 2 and 1. Hence the protocol is communication efficient. The simulation results shows that, proposed protocol requires less number of encryptions/ decryptions on KMS side as well as group member side. This protocol is storage efficient compared to other schemes. For a large group, this protocol can be combined with cluster based or tree based protocols.

Table 4. Comparison of proposed protocol with other schemes

Protocol	Storage Cost (Number of keys stored)		Computation cost		Communication cost (Number of rekeying messages)	
	KMS	GM	KMS	GM	Join	Leave
LKH	$2n-1$	$\log_2 n+1$	$(\log_2 n)E+(\log_2 n)K_g$	$(\log_2 n)D$	$2\log_2 n$	$2\log_2 n$
Centralized Flat Table	$2I+1$	$I+1$	$(I+1)E+(I+1)K_g$	$(I+1)D$	$I+1$	$I+1$
GKMP	$n+1$	1	$(n+1)E+K_g$	D	1	n
SKTPCRT	$2n$	2	$nE+nM_I+nM+K_g$	M_d+D	1	1
Proposed Protocol	$\boldsymbol{n}$**+1**	$\boldsymbol{n}$	**1*E*+*n*(o_1+o_2)**	**1*D*+*n*(o_1+o_2)**	**1**	**1**

Notations used in Table 4.

- n is number of group members
- K_g is computation cost of key generation using standard key generation algorithm
- E/D is Encryption/decryption
- M is multiplication, H is hash operation
- M_I is multiplicative inverse operation
- M_d is modular division to find the remainder
- I is number of bits required to represent the group ID $\cong \lceil \log_2 n \rceil$
- $o_1, o_2 = \{+, \vee, \oplus, \ll_l\}$

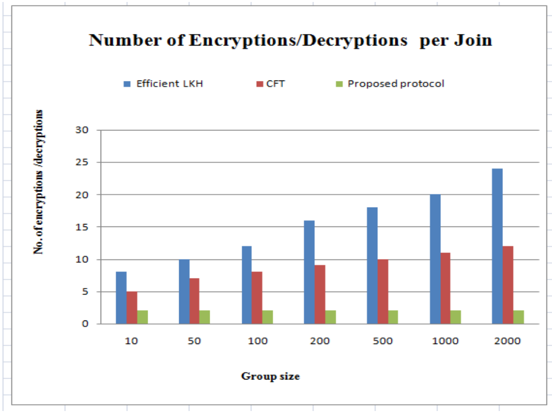


Fig.9. Total number of encryptions/decryptions per join

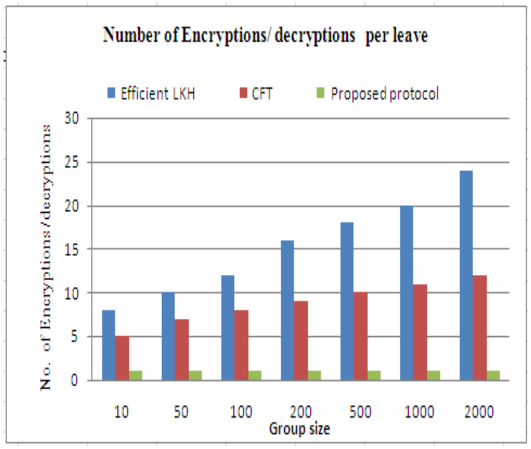


Fig.10. Total number of encryptions/decryptions by KMS per leave

REFERENCES

[1] H. Harney and C. Muckenhirn, A *Group Key Management Protocol (GKMP)* RFC 2093,1997

[2] D. Wallner, E. Harder, and R. Agee, “Key Management for Multicast: Issues and Architectures”, *Internet draft*, September 1998.

[3] C. K. Wong, M. Gouda, and S. S. Lam, “Secure group communications using key graphs”, *IEEE/ACM Trans. on Networking*, vol.8 (1): pp-16-30, 2000

[4] A. T. Sherman and D.A. McGrew, “Key establishment in large dynamic groups using one-way function trees”, IEEE transactions on Software Engineering, vol. 29, no. 5, pp. 444-458, 2003.

[5] L. Jing and B. Yang, “Collusion-Resistant Multicast Key Distribution Based on Homomorphic One-Way Function

Trees" IEEE Trans. On Information Forensics and Security, vol. 6, no 3, Sept. 2011 pp-980-99

[6] M. Moyer, J. Rao, and P. Rohatgi, "Maintaining Balanced Key Trees for Secure Multicast," Internet *draft, draft-irtf-smug-keytree- balance*-00.txt, June 1999

[7] O. Rodeh, K.P. Birman, and D. Dolev, "Using AVL Trees for Fault Tolerant Group Key Management," *Int'l J. Information Security*, pp. 84-99, Nov. 2001.

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

[9] H. Lu, "A Novel High-Order Tree for Secure Multicast Key Management", *IEEE transaction on Computers*, vol. 54, no. 2, Feb.2005 pp 214-224.

[10] D. W. Kwak, S. J. Lee, J. Kim, E. Jung, "An Efficient LKH Tree Balancing Algorithm for group key management", *IEEE Communications Letters*, vol. 10, Issue3, pp.222-224, March 2006

[11] M. Waldvogel, G. Caronni, D. Sun, et al., "The Versa Key Framework: Versatile Group Key Management", *IEEE Journal on Selected Areas in Communications*, vol.17 (9), pp: 1614–1631, September1999.

[12] G. H. Chiou and W. T. Chen, "Secure Broadcast using secure lock," *IEEE Trans. on Software Engineering*, vol. 15, no. 8, pp. 929–934, Aug. 1989.

[13] M. Y. Joshi and R. S. Bichkar, "Scalable Key Transport Protocol Using Chinese Remainder Theorem", *The Proceedings of International symposium on Security in Computers and Communications (SSCC) 2013, Mysore, India , pp. 397-402*

[14] R. Song, L. Korba, O. George, and M. Yee, "A Scalable Group Key Management Protocol", *IEEE Communications Letters*, vol. 12, no. 7, pp. 541-543 ,July 2008

[15] R. Canetti, T. Malkin, K. Nissim, "Efficient Communication Storage tradeoffs for multicast encryption" LNCS 1592(1999) Advances in Cryptology – EUROCRYPT'99.

[16] S. Rafaeli and D. Hutchsion, "A Survey of Key Management for Secure Group Communication", ACM *Computing Survey*, Vol. 35, No.3, pp 309-329, 2003.

[17] B. Bezawada, and S.S. Kulkarni, "Balancing Revocation and Storage Trade-Offs in Secure Group Communication", *IEEE Trans. on Dependable and Secure Computing*, vol. 8, no. 1, 2011.

[18] E. Rescorla, *SSL and TLS: Designing and Building Secure Systems*, Reading, MA: Addison-Wesley, 2001.

[19] P. Vijayakumar, S. Bose, and A. Kannan, "Chinese remainder Theorem based centralised group key management for secure multicast communication" *IET Information Security*, Vol. 8, Issue. 3, pp. 179–187, May 2014

[20] W. Stalling "*Cryptography and Network Security, Principals and Practices*" 5th Edition, Prentice Hall, ISBN 13: 978-0-13-609704-4

Authors Biographies

Manisha Y. Joshi has completed her B. E. Electronics and M. E. Electronics from Marathwada University, Aurangabad in 1994 and 1999 respectively. She is working as Faculty in Computer Science and Engineering in MGMs College of Engineering, Nanded since August 1999. She is currently pursuing Ph. D. in Swami Ramanand Teerth Marathwada University, Nanded. Her areas of interest are cryptography, network security and secure group communications.

Rajankumar S. Bichkar has completed his B. E. Electronics and M. E. Electronics from SGGS College of Engineering and Technology, Nanded (then affiliated to Marathwada University, Aurangabad) in 1986 and 1991 respectively. He has completed his Ph. D. from IIT, Kharagpur in 2000. He has worked as Lecturer, Asst. Prof. and Professor (till 2007) at SGGS Institute of Engineering and Technology, Vishnupuri, Nanded and is currently working as Professor (E&TC) and Dean (R&D) in GH Raisoni College of Engineering & Management, Pune since 2008. He has to his credit 3 books (on Basic and C Programming) and more than 30 papers in Conferences and Journals.

How to cite this paper: Manisha Y. Joshi, Rajankumar S. Bichkar,"A Centralized Key Table based Communication Efficient Group Key Management Protocol", IJCNIS, vol.7, no.8, pp.49-55, 2015.DOI: 10.5815/ijcnis.2015.08.06