A New Secure Multicast Key Distribution Protocol Using Combinatorial Boolean Approach

Mohamed M. Nasreldin Rasslan, Yasser H. Dakroury, and Heba K. Aslan
(Corresponding author: Heba K. Aslan)

Informatics Department of Electronics Research Institute, Cairo, Egypt
Etahrir Street Dokki Cairo Egypt (Email: hebaaslan@yahoo.com)

(Received Nov. 26, 2006; revised June 15, 2007; and accepted Dec. 5, 2007)

Abstract

In the present paper, we propose a new protocol for a scalable multicast key distribution protocol. The proposed protocol is based on Key Management using Boolean Function Minimization (KM-BFM) technique. It is considered one of the best solutions proposed for solving the scalability of multicast security protocols depending on a centralized manager. Instead of using one tree as in KM-BFM, the members are divided into a number of subgroup trees. A comparison between KM-BFM and other centralized protocols is detailed. The comparison shows that KM-BFM protocol has the lowest communication overhead. Furthermore, it has the lowest storage at the security manager. Then, a comparison between the proposed protocol and KM-BFM protocol is given. The comparison shows that the proposed protocol has lower storage requirements. Further, it achieves a lower communication overhead in case of a single member leave and a comparable communication overhead in case of multiple leaves.

Keywords: Boolean approaches, combinatorial approaches and centralized approaches, key distribution, multicast security

1 Introduction

Multicast communication as defined in [10] is an efficient mean of distributing data to a group of participants. Multicast communication has many economical and military applications, some of them need to deal with groups of users and keep each group protected from other group access. Group key distribution techniques are used in many such applications like internet video transmissions, stock quotes, news feeds, life multi-part conferencing and battlefield [3]. Multicast communication needs mechanism(s) to control access to the transmitted data and protect the group communication from illegitimate members. Multicast security suggests encryption to protect messages exchanged among group members, so generating, distributing and updating of the cryptographic keys become hot issues especially when group membership is highly dynamic.

Group key must be updated with the group membership changes to prevent a new member from deciphering messages exchanged before it joins the group; this is defined as backward secrecy [3]. Group key revocation in case of one member joins or multiple members join could be achieved by sending the new group key to the old group members encrypted with the old group key. Also, group key must be updated with the group membership changes to prevent an old member (leaved or expelled) from deciphering current and future group communication which is defined as forward secrecy [3]. Group key revocation, when one member leaves or multiple members leave, is more complicated than in case of join because of the disclosure of the old group key. The old group key is known to the leaving member(s) so there is a need to re-key the group using valid key(s) in a scalable way. The trivial scheme for re-keying a group of n members is through using individual secret key shared between the Key Distribution Center KDC and each member. This is not a simple or scalable method and consumed large bandwidth especially for large group with high membership changes; furthermore it takes more time and needs more resources per hosts than using multicasting to re-key the group. In literature, there are many group key distribution protocols. In the next subsection, a brief summary of previous key distribution protocols is defined.

1.1 Related Work

According to [14], the classification of multicast key distribution protocols is as follows:

Centralized group key management protocols: A single entity is employed for controlling the whole group; hence a group key management protocol seeks to minimize storage requirements, computational power on both client and server sides, and bandwidth utilization. Although the centralized approach has a problem of a single point of failure, some applications like stock quotes

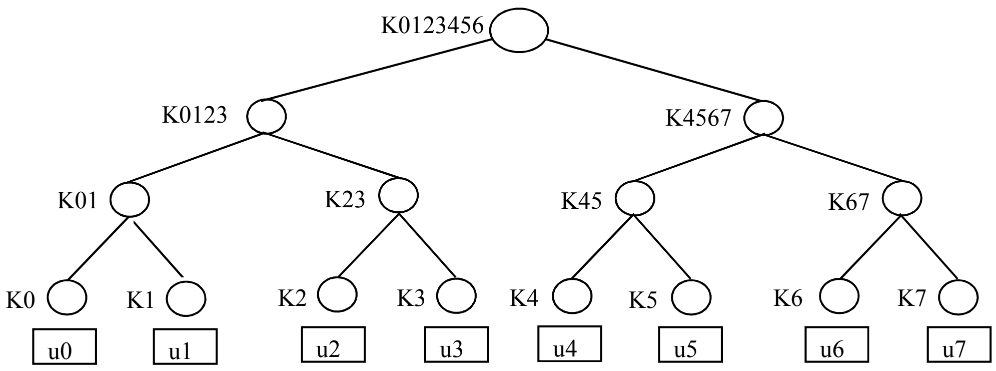

Figure 1: Logical key hierarchy tree

are still centralized. To overcome this problem, a mirror of the centralized entity could be used to provide fault tolerant and/or load sharing. Some examples of centralized group key protocols are: Logical Key Hierarchy (LKH) [21], One-Way Function Trees (OFT) [11] and Key Management using Boolean Function Minimization (KM-BFM) technique [4]. The centralized approaches are generally based on the idea of LKH where a key distribution center maintains a key tree as shown in Figure 1. Each node in the tree represents a symmetric key. The leaves represent the group members. Each member knows all the symmetric keys from its leaf to the root. For example, U_0 knows K_0, K_{01}, K_{0123} and the group key $K_{01234567}$. If U_7 leaves the group, to maintain forward secrecy, the following keys must be changed: $K67$, K_{4567} and $K_{01234567}$. The key server generates K_{67}(new), K_{4567}(new) and $K_{01234567}$(new) and broadcasts the following message: $\{K_{67}$(new), K_{4567}(new), $K_{01234567}$(new)$\}K_6$, $\{K_{4567}$(new), $K_{01234567}$(new)$\}K_{45}$, $\{K_{01234567}$(new)$\}K_{0123}$. U_6 obtains the new keys by decrypting the part encrypted by K_6, U_4 and U_5 obtain the new keys by decrypting the part encrypted by K_{45} and finally users: U_1, U_2, U_3 and U_4 obtain the group key by decrypting the part encrypted by K_{0123}. Therefore, for a group of m members, when a single member leaves, the key server updates the group keys using a message of length 2 $\log_2 m$ keys. In LKH protocols, the key server stores $(2m + 1)$ symmetric keys and each member stores $(\log_2 m + 1)$ symmetric keys.

In order to reduce both storage at the key server and the length of update message, Chang et al. [4] proposed a key distribution protocol which we named Key Management using Boolean Function Minimization (KM-BFM) protocol. KM-BFM protocol is based on the idea of using only two keys in each level as shown in Figure 2. Each member knows the symmetric keys along its path to the root. To reduce the number of messages to be exchanged and to decrease the number of encryption operations needed to be performed by the key server, one-way functions are used. If U_0 leaves the group, the following keys must be changed: SK, $\bar{K_0}$, $\bar{K_1}$ and $\bar{K_2}$. The key server generates SK(new) and broadcasts the following message: SK(new)K_0, SK(new)K_1, SK(new)K_2. Upon receiving the message, the remaining users can obtain SK(new) by decrypting one part of the message. Then, the new keys: $\bar{K_0}$, $\bar{K_1}$ and $\bar{K_2}$ are calculated using one-way functions of the old keys and the new group key, i.e. $\bar{K_0}$(new)=f(SK(new),$\bar{K_0}$), $\bar{K_1}$(new)=f(SK(new),$\bar{K_1}$) and $\bar{K_2}$(new)=f(SK(new),$\bar{K_2}$). Therefore, for a group of m members, when a member leaves, the key server updates the group keys using a message of length $\log_2 m$ keys. In KM-BFM protocol, the key server stores $2(\log_2 m + 1)$ keys and each member stores $(\log_2 m + 1)$ keys. Therefore, KM-BFM protocol achieves lower communication and computation overheads compared to LKH protocols. Further enhancement to both communication and computation overheads could be achieved in case of multiple leaves as mentioned in [4]. In KM-BFM, the worst case occurs when certain $m/2$ members leave the group. Therefore, the key server has to update each remaining member individually and the length of update message is $m/2$ keys.

Decentralized key management protocols: The management of a large group is divided among subgroup managers, trying to minimize the problem of concentrating the work in a single manager. These protocols need more trusted nodes and suffer from encryptions and decryptions processes between subgroup managers. Some examples of decentralized protocols are: Scalable Multicast Key Distribution using Core Based Tree (CBT) [1], Iolus [12], Dual-Encryption Protocol (DEP) [5] and Kronos [16].

Distributed key management protocols: There is no explicit manager, and the members themselves do the key generation. All members can perform access control and the generation of the key can be rather contributory, meaning that all members contribute some information to generate the group key, or done by one of the members. The distributed protocols have a scalability problem in case of key update, since they require performing large

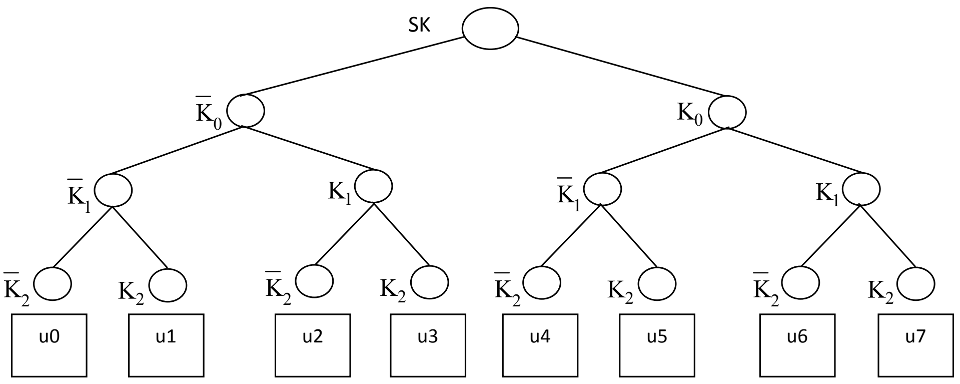

Figure 2: Key management using boolean function tree

computations and they are characterized by large communication overheads. Further, they need all group members to have powerful resources. Some examples of distributed key management protocols are: Octopus Protocol [2], Distributed Logical Key Hierarchy [15] and Diffie-Hellman Logical Key Hierarchy [8]. In the following subsection, an overview of the proposed protocol is given.

1.2 Overview of the Proposed Protocol

According to [14], the efficiency of the protocol can be measured by storage requirements (number of stored keys at both key server and each member), size of update messages, backward and forward secrecy and collusion resistance (evicted members must not be able to work together and share their individual piece of information to regain access to the group key). Therefore, an ideal multicast key distribution protocol is the one that achieves the lowest storage at both the server and the group members. Also, it has the lowest update message length. Furthermore, it preserves both forward and backward secrecy and lowers the collusion attack. In the present paper, we propose a new protocol for multicast key distribution. The aim of the new protocol is to enhance the scalability of the group key management protocol that is used in a centralized system through minimizing the requirements of group manager and group members. The proposed protocol is based on KM-BFM protocol. Instead of using one tree as in KM-BFM, the members are divided into a number of subgroup trees. The group manager holds n key pairs and each group member holds y keys. The number of subgroups is (nCy), where C represents a combination function. As will be shown, the proposed protocol achieves a lower storage at both the group manager and the group members compared to KM-BFM protocol. Also, it has a lower update message length in case of a single member leave and a comparable update message length in case of multiple leaves. Furthermore, the probability of conducting a successful collusion attack in the proposed protocol is less than that proposed in KM-BFM protocol. This is due to the fact that, in the proposed protocol, each member knows less number of keys than in KM-BFM protocol. It has to be noted that the authentication problem is not addressed in the present paper. To achieve authenticity, protocols proposed in [13] and [6] could be used. The paper is organized as follows: in the next section, description of the proposed protocol is detailed. Next, in the Section 3, comparison of the proposed protocol with other centralized protocols is depicted. Finally, the paper concludes in Section 4.

2 Proposed Multicast Key Distribution Protocol

The proposed protocol is based on the idea of KM-BFM protocol. In KM-BFM, all the group members are organized in one tree hierarchy controlled by a group manager. The group manager holds m pairs of auxiliary keys and each group member holds m keys, where m represents the tree length and the number of group members equals 2^m. On the other hand, in our approach, the group members are divided into 'nCy' subgroups (C represents a combination function and equals $\frac{n!}{(n-y)!(y)!}$) managed by a group manager, where the manager holds n pairs of auxiliary keys and y represents the length of each subgroup tree ($1 \leq y \leq n-1$). Each group member holds 'y' auxiliary keys. For the same number of group members, 'n' is less than 'm' as will be shown later. In the following subsections description of the proposed protocol will be illustrated.

2.1 General Notations and Assumptions

For the operation of the proposed protocol the following assumptions are made: the group manager holds n key pairs and each member holds y keys, where y represents subgroup tree height. The number of subgroup equals nCy. Each subgroup contains 2^y members. Therefore, the total number of group members 'N' equals $nCy \times 2^y$ members. In our approach, each group member is associated with a unique binary string of length y. This binary string is the unique identifier UID for each member. We

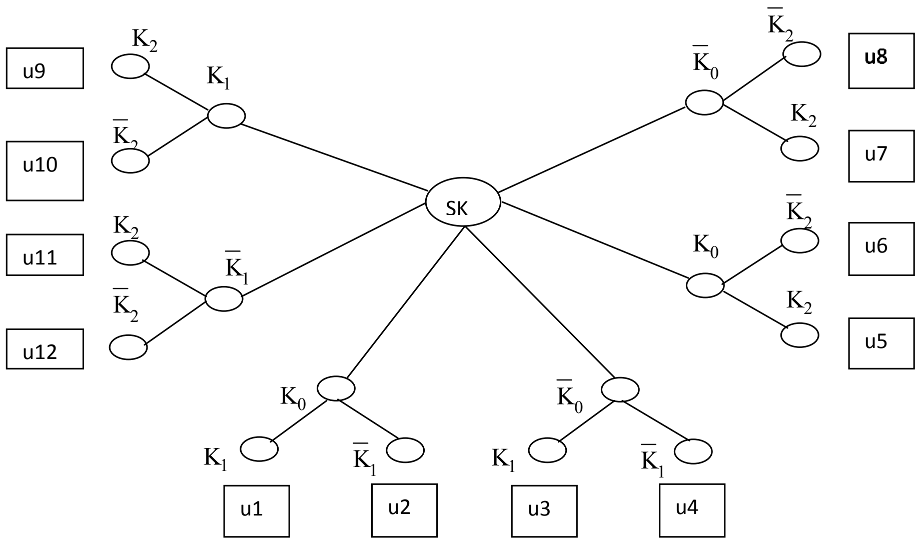

Figure 3: Proposed protocol subgroup key hierarchy for $n = 3$ and $y = 2$

can write the member UID as X_0, ...,X_{y-2}, X_{y-1}, where X_i can be either 1 or 0. As mentioned above each group member holds y auxiliary keys. Let these auxiliary keys represented by the following: $K_0, \dots, K_{y-2}, K_{y-1}$, where K_i written as K_i if $X_i = 1$ and $\bar{X_i}$ if X_i=0. It has to be noted that the values of K_i and $\bar{K_i}$ are not complements of each other, they are two unrelated keys. Each key pair corresponds to a bit in the UID. These y keys are drawn from a set of n key pairs which are known to the group manager. Figure 3 shows an example for 2-bits UID. In this figure, $y = 2$ and $n = 3$. Subsequently, the group members will be equal to (3C2) * (2^2) = 12, each group member holds 2 keys and the group manager holds 6 keys. On the other hand, for KM-BFM protocol, a system containing 12 users must use 4-bits user ID. Therefore, the manager holds 8 keys and each group member holds 4 keys. This implies that our approach leads to a smaller storage at both the group manager and group members. In Figure 3, the round nodes represent the auxiliary keys. Each member possesses all the keys on the branch from the leaf to the root of the tree. For example u_2 possesses the auxiliary keys $\bar{K_1}$ and K_0 in addition to the group key (current session key) which is known to all group members. In order to obtain the value of y which maximizes the total number of group members 'N', we differentiate N with respect to y. First, N is given by:

$$\begin{aligned} N &= (nCy)(2^y) \\ &= \frac{n!}{(n-y)!(y)!}2^y \\ &= \frac{\Gamma(n+1)}{\Gamma(n-y+1)\Gamma(y+1)}2^y, \qquad (1) \end{aligned}$$

where, $n!$ is the factorial of n [9] and $\Gamma(n)$ is the Gamma Function, $n! = \Gamma(n+1)$ [9]. Then, '$\frac{\delta N}{\delta y}$' is given in the following equation:

$$\begin{aligned} \frac{\partial N}{\partial y} &= \frac{\partial}{\partial y}\left[\frac{\Gamma(n+1)2^y}{\Gamma(n-y+1)\Gamma(y+1)}\right] \\ &= \frac{2^y\Gamma(n+1)}{\Gamma(n+1-y)\Gamma(y+1)} \times \\ &\quad [\Psi(n+1-y) - \Psi(y+1) + \ln(2)], \qquad (2) \end{aligned}$$

where, $\Psi(n)$: $\Psi(n)$ function is the logarithmic derivative of gamma function [20]. The critical points are obtained by finding the values of y which make $\frac{\delta N}{\delta y}$ equals zero [20]. Then, the second derivative $\frac{\delta^2 N}{\delta y^2}$ is calculated to check the value of y that leads to maximize N [18]. $\frac{\delta^2 N}{\delta y^2}$ is given by:

$$\begin{aligned} \frac{\partial^2 N}{\partial y^2} &= \frac{\partial}{\partial y}[\frac{2^y\Gamma(n+1)}{\Gamma(n+1-y)\Gamma(y+1)} \times \\ &\quad [\Psi(n+1-y) - \Psi(y+1) + \ln(2)]] \\ &= \frac{\Gamma(n+1)2^y}{\Gamma(n+1-y)\Gamma(y+1)} * \\ &\quad [-(n+1-y)\Psi^{(1)} - (y+1)\Psi^{(1)} \\ &\quad +((n+1-y)\Psi - \Psi(y+1) + \ln(2))^2]. \qquad (3) \end{aligned}$$

Equations (2) and (3) are used to obtain Table 1 which shows a comparison between KM-BFM protocol and the proposed protocol. In Table 1, we see the effect of combinations in reducing the storage of auxiliary keys at both the group manager and group members. As mentioned above each group member holds y keys and the group manager holds n key pairs. On the other hand, in KM-BFM, for the same number of users, each group member holds m keys and the group manager holds m key pairs. Table 1 shows that m is greater than both n and y. Furthermore, for the same storage at the group manager, the number of group members concerning our protocol is greater than the number of group members in KM-BFM protocol.

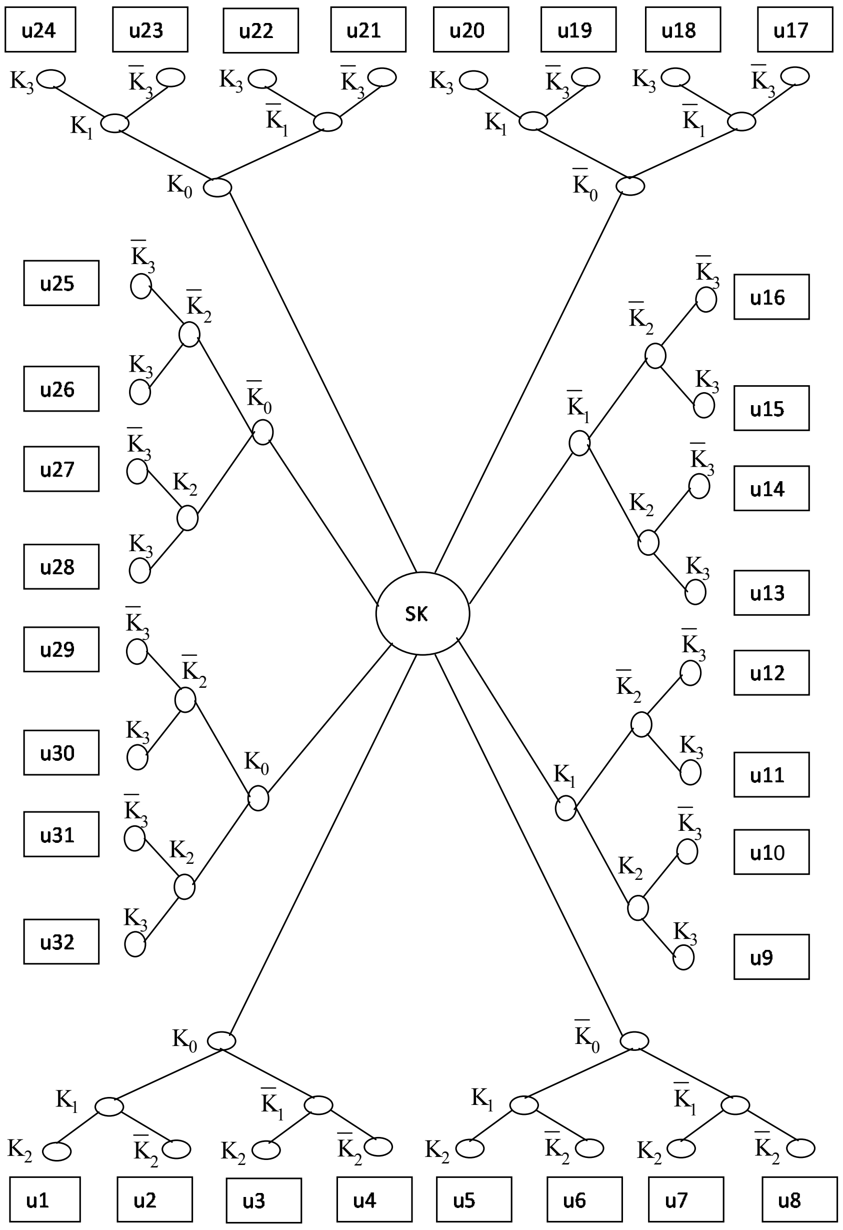

Figure 4: Proposed protocol subgroup key hierarchy for $n = 4$ and $y = 3$

Table 1: Comparison of the proposed protocol with KM-BFM protocol

n	y	N	2^n	m	2^m
2	1	4	4	2	4
3	2	12	8	4	16
4	3	32	16	5	32
10	7	15,360	1024	14	16,384
15	10	3,075,072	32,768	22	4,194,304
25	17	1.4176e+011	33,554,432	38	2.7488e+11
30	20	3.1504e+013	1.0737e+9	45	3.5184e+13
40	27	1.6151e+018	1.0995e+12	61	2.3058e+18
45	30	3.7030e+020	3.5184e+13	69	5.9030e+20

* where, n: number of auxiliary key pairs hold by the group manager in the proposed protocol.
y: number of auxiliary keys hold by each group member in the proposed protocol.
N: number of group members in the proposed protocol.
2^n: number of group members in KM-BFM protocol assuming the same storage at the group manager.
m: number of auxiliary key pairs hold by the group manager in KM-BFM protocol in order to cover the same group size as the proposed protocol.

2.2 Member Leave

When a member leaves the multicast group or the group manager deletes it (because its subscription has expired), a new session key needs to be established by the group manager and distributed to the remaining members in the multicast group. Let the current session key be $SK(t)$, where t is the index of the current round, and the new session key is $SK(t+1)$. When a member leaves, the group manager computes $SK(t+1)$ and encrypts it using the auxiliary keys set after excluding the known auxiliary keys for the leaving member. Figure 4 represents a system with 32 members. The group manager stores 8 auxiliary keys in addition to the session key. Each member in the group stores 3 auxiliary keys in addition to the session key. In this example, $n = 4$ and $y = 3$, therefore we have 4 subgroups ($4C3$) each subgroup has a height of 3. Due to the use of binary notation each subgroup has 2^3 members, thus the total number of the group members is $4C3 \times 2^3 = 32$. Assume that U30 is deleted form the group or leaves the group. U30 possesses keys K_0, $\bar{K_2}$ and K_3. The new session key $SK(t+1)$ is encrypted using 5 different keys which are not known to U30. These keys are $\bar{K_0}$, K_2, $\bar{K_3}$, K_1 and $\bar{K_1}$. Next the group manager sends the output of encryption in one multicast message containing $[SK(t+1)]\bar{K_0}$, $[SK(t+1)]K_2$, $[SK(t+1)]\bar{K_3}$, $[SK(t+1)]K_1$ and $[SK(t+1)]\bar{K_1}$. Every remaining member in the group can decrypt at least one message. This is due to the fact that every remaining member's auxiliary key set differs at least in one auxiliary key from the leaving member's auxiliary key set. Therefore, the new session key can be decrypted by members from U_1 to U_{24} using K_1 or $\bar{K_1}$. For members from U_{25} to U_{32} except U_{30} the new session key can be decrypted using $\bar{K_0}$, K_2 or $\bar{K_3}$. Since U_{30} does not know the keys used in encryption, therefore it will not be able to obtain the new session key. For a group of N members, after deletion of a single member, y auxiliary keys of the $2n$ auxiliary keys are excluded when the manager updates the session key.

Subsequently, the length of update message that need to be sent out to update the session key after the deleting a single member is $2n - y$, with each message encrypted with one of the remaining keys. On the other hand, the number of messages that need to be encrypted and sent in KM-BFM protocol is $\log_2(N)$ [4]. After distributing the new session key $SK(t+1)$, auxiliary keys possessed by the leaving member needs to be updated to make sure that the leaving member cannot use its auxiliary keys to decrypt future session key updates. In the above example, $K_0(t)$, $\bar{K_2}(t)$ and $K_3(t)$ must be changed. The new keys are calculated as follows:

$$\begin{aligned} K_0(t+1) &= f(K_0(t), SK(t+1)), \\ K_2(t+1) &= f(\bar{K_2}(t), SK(t+1)), \\ K_3(t+1) &= f(K_3(t), SK(t+1)), \end{aligned}$$

where f is a one-way function which has the old key value and the new session key as inputs, and it outputs the new key value. In the proposed protocol, the group manager calculates y one-way functions, while in KM-BFM protocol, a group manager calculates m one-way functions. Further, a group member calculates at most $(y-1)$ one-way functions, while in KM-BFM protocol, a group member calculates at most $(m-1)$ one-way functions. In the next subsection, multiple leaves will be explained.

2.3 Multiple Leaves

When several members are deleted or left in the same round, it is better to aggregate the deletion of all members from the group instead of deleting each member individually. In general, consider a set of members, $U = \{U_1, U_2, \ldots, U_N\}$, where $N = nCy \times 2^y$. Each member knows y auxiliary keys and its combinations. For example, in Figure 4, U_{30} knows the following patterns; K_0, $\bar{K}_2$, K_3, $K_0\bar{K}_2$, K_0K_3, $\bar{K}_2K_3$ and $K_0\bar{K}_2K_3$. The new session key $SK(t+1)$ can be encrypted using composite key if all single auxiliary keys are invalidated due to deletion of members. The composite key is derived from a one-way function of two keys or more to avoid the extra cost of multiple encryptions. To update SK, first, we use single auxiliary key pattern to cover all the remaining members. When the single auxiliary key pattern does not cover all the remaining members, we use pattern of double keys and so on until cover all remaining members.

In the following, we present a systematic approach to solve the problem of deleting members in the same round and to find the minimum number of valid auxiliary keys that cover all remaining members. Given a set of subsets $P = \{P_1, P_2, \ldots, P_s\}$ of the universal set $U = \{U_1, U_2, \ldots, U_N\}$, where s is number of auxiliary key patterns $s = \sum_{m=1}^{y}(nCm)(2^m)$. Each P_i (key pattern) covers subset of the users set U. We aim to find multiple subsets T_i of P such that $\cup T_i$ covers U_r, where $\cup T_i$ represents the union of all subsets T_i and U_r is the remaining members set after deleting or leaving of the member(s). Minimum set cover is the problem of finding the minimum sub-collection (cover) of subsets T_i whose union gives all the elements of the main set U_r. The elements of the given subsets Pi should cover all the elements of the main set U_r. The set-covering problem, basically, is an abstraction of many commonly arising combinatorial problem and it is an optimization problem that models many resources selection problem. The problem is solved approximately by Greedy paradigm [7]. In Greedy approach, the next subset selected for the minimum set cover is the subset that covers the most uncovered elements. To find such a subset, the effective weights (number of uncovered elements) of the subsets are calculated in each iteration until the effective weights of all subsets are zero. Greedy approach for set covering problem does not always guarantee an optimal solution. The more efficient approach would be Backtracking approach [18] which always guarantees an optimal solution, but this is achieved with high computations and typically it is not worth the computational expense.

We construct a table having all remaining members as columns and all auxiliary key patterns as rows, then use Greedy heuristic algorithm to search through our table. Begin by placing the largest subset in the set cover, and then mark all its elements as covered. Then repeat for all remaining subsets containing the largest number of uncovered elements, until all elements are covered. Steps to search through the table are given below:

Step 1: Construct a table having all members as columns and all auxiliary key patterns as rows.

Step 2: Put 1 in cell (i, j) to indicate that pattern P_i covers member U_j.

Step 3: When member(s) U_j leaves the group, delete all row(s) that have 1 in column(s) j, then delete all column(s) j.

Step 4: Select a row which contains maximum number of 1's (cover maximum members). In the case where two rows have the same number of 1's, choose the row which contains a user with the least column index. Then, put this Pi to be the subset T_k, where k has an initial value equals to 1.

Step 5: Remove all members U_j that have 1 in row P_i. Then, remove row P_i.

Step 6: If $\cup T_i \neq U_r$, increase k by one and go to Step 4.

For example, in Figure 3 assume that U_3 and U_{12} leaved the group, $n = 3$ and $y = 2$. The universal set $U = \{U_1, U_2, U_3, \ldots, U_{11}, U_{12}\}$, the remaining members set $U_r = \{U_1, U_2, U_4, U_5, U_6, U_7, U_8, U_9, U_{10}, U_{11}\}$ and $s = \sum_{m=1}^{2}(3Cm)(2^m)$=18.

Let pattern P_1 as K_0, P_2 as $\bar{K}_0$, P_3 as K_1, P_4 as $\bar{K}_1$, P_5 as K_2, P_6 as $\bar{K}_2$, P_7 as K_0K_1, P_8 as $K_0\bar{K}_1$, P_9 as $\bar{K}_0K_1$, P_{10} as $\bar{K}_0\bar{K}_1$, P_{11} as K_0K_2, P_{12} as $K_0\bar{K}_2$, P_{13} as $\bar{K}_0K_2$, P_{14} as $\bar{K}_0\bar{K}_2$, P_{15} as K_1K_2, P_{16} as $K_1\bar{K}_2$, P_{17} as $\bar{K}_1K_2$ and P_{18} as $\bar{K}_1\bar{K}_2$.

From Figure 3, each pattern covers the following users: $P1 = \{U_1, U_2, U_5, U_6\}$, P_2=$\{U_3$, U_4, U_7, $U_8\}$, P_3=$\{U_1$, U_3, U_9, $U_{10}\}$, P_4=$\{U_2$, U_4, U_{11}, $U_{12}\}$, P_5=$\{U_5$, U_7, U_9, $U_{11}\}$, P_6=$\{U_6$, U_8, U_{10}, $U_{12}\}$, P_7=U_1, P_8=U_2, P_9=U_3, P_{10}=U_4, P_{11}=U_5, P_{12}=U_6, P_{13}=U_7, P_{14}=U_8, P_{15}=U_9, P_{16}=U_{10}, P_{17}=U_{11} and P_{18}=U_{12}. Table 2 represents Steps 1 and 2.

Applying Step 3 after removing both U_3 and U_{12} will lead to Table 3.

Next, applying Step 4 leads to Table 4.

Therefore, the first subset is $T_1 = P_1$ =$\{U_1$, U_2, U_5, $U_6\}$. After removing users contained in P_1, we will obtain Table 5.

Applying Step 4 again, we will obtain the second subset $T_2 = P_5$=$\{U_7$, U_9, $U_{11}\}$. After removing users contained in P_5, we will obtain Table 6.

By repeating the algorithm, we obtain: $T_3 = P_{10} = U_4$, $T_4 = P_{14} = U_8$ and $T_5 = P_{16} = U_{10}$.

Finally, union of T_1, T_2, T_3, T_4 and T_5 will cover all the remaining group members. To update the session key in the above example, the group manager needs to encrypt the new session key using K_0, K_2, $\bar{K}_0\bar{K}_1$, $\bar{K}_0\bar{K}_2$, and $K_1\bar{K}_2$.

For scalability and efficiency reasons, it is desired that re-keying requires a minimum number of messages sent to the group and/or that a minimum number of encryption operations are performed. This is achieved by encrypting re-keying information with keys common to subsets of the remaining members. In our approach, number of updating messages depends not only on number of leaving members, but also on their positions in the multicast group. Deleting members from the same subgroup needs less number of updating messages than deleting them from different subgroups in the multicast group. The worst case in our approach occurs when a certain sequence of members leave the group with knowledge of all

Table 2: All key patterns and each corresponding user

	U_1	U_2	U_3	U_4	U_5	U_6	U_7	U_8	U_9	U_{10}	U_{11}	U_{12}
P_1	1	1			1	1						
P_2			1	1			1	1				
P_3	1		1						1	1		
P_4		1		1							1	1
P_5					1		1		1		1	
P_6						1		1		1		1
P_7	1											
P_8		1										
P_9			1									
P_{10}				1								
P_{11}					1							
P_{12}						1						
P_{13}							1					
P_{14}								1				
P_{15}									1			
P_{16}										1		
P_{17}											1	
P_{18}												1

Table 3: Removing U_3 and U_{12}

	U_1	U_2	U_4	U_5	U_6	U_7	U_8	U_9	U_{10}	U_{11}
P_1	1	1		1	1					
P_5				1		1		1		1
P_7	1									
P_8		1								
P_{10}			1							
P_{11}				1						
P_{12}					1					
P_{13}						1				
P_{14}							1			
P_{15}								1		
P_{16}									1	
P_{17}										1

Table 4: Applying Step 4

	U_1	U_2	U_4	U_5	U_6	U_7	U_8	U_9	U_{10}	U_{11}
P_1	1	1		1	1					
P_5				1		1		1		1
P_7	1									
P_8		1								
P_{10}			1							
P_{11}				1						
P_{12}					1					
P_{13}						1				
P_{14}							1			
P_{15}								1		
P_{16}									1	
P_{17}										1

Table 5: Removing users contained in P_1

	U_4	U_7	U_8	U_9	U_{10}	U_{11}
P_5		1		1		1
P_7						
P_8						
P_{10}	1					
P_{11}						
P_{12}						
P_{13}		1				
P_{14}			1			
P_{15}				1		
P_{16}					1	
P_{17}						1

Table 6: Removing users contained in P_5

	U_4	U_8	U_{10}
P_7			
P_8			
P_{10}	1		
P_{11}			
P_{12}			
P_{13}			
P_{14}		1	
P_{15}			
P_{16}			1
P_{17}			

auxiliary key patterns. This forces the group manager to update the session key for each remaining member due to invalidation of key patterns with length l, where $l = y-1$. The number of key patterns of length l from n auxiliary keys using Boolean notations 'N_p' is given by:

$$N_p = nCl * 2^l.$$

Further each member in the group knows yCl key patterns of length l.Therefore, the number of leaving members that leads to the worst case 'W' is given by the following equation:

$$W = \left\lceil \frac{Np}{yCl} \right\rceil = \left\lceil \frac{nCl \times 2^l}{yCl} \right\rceil = \left\lceil \frac{nC(y-1) \times 2^{y-1}}{y} \right\rceil.$$

Therefore, the number of update messages 'M' is given by:

$$M = N - W. \tag{4}$$

Where N is the total number of group members.

In Figure 4, $W = \frac{4C2 \times 2^2}{3C2} = 8$. Therefore, there is a sequence of 8 members, when they leave, the group manager updates the remaining members individually. The probability of this case equals $(1/(32C8)) \times 100 = 9.5 \times 10^{-4}\%$. Although this probability is small and goes very small as the group size goes larger, we can limit the number of messages by restricting the group size to a certain value instead of the maximum value N. If we desire the number of messages to be D, we can limit the number of members to be $W + D$. This reduces the group size but ensures that number of update messages will not exceed D messages. In the next subsection, analysis of the proposed protocol is given.

2.4 Analysis of the Proposed Protocol

The efficiency of a protocol can be measured by: storage requirements, size of update messages, backward and forward secrecy and collusion resistance. The storage requirements and the size of update messages will be compared with other centralized protocols in the next section. In the following paragraphs, we analyze if the proposed protocol achieves both backward and forward secrecy. Also, we analyze if it resists collusion attacks.

Backward secrecy: backward secrecy means that the group key must be updated when a new member joins the group. This is to prevent it from deciphering messages exchanged before it joins the group. Assume U_5 in Figure 3 joins the group, K_0, K_2 and SK must be changed to K_0(new), *place*K_2(new) and SK(new). The key server generates SK(new) and broadcasts the following message: $\{SK(\text{new})\}SK$. Upon receiving the message, all group members except U_5 can obtain SK(new) by decrypting the message using the old value of SK. Then, the new keys: K_0(new) and K_2(new) are calculated using one-way functions of the old keys and the new group key, i.e. $K_0(\text{new}){=}f(SK(\text{new}),K_0)$, $K_2(\text{new}){=}f(SK(\text{new}), K_2)$. Since U_5 does not know the old values of K_0, K_2 and SK, therefore it cannot obtain or calculate K_0(new), K_2(new) and SK(new). This means that it could not access past group communications.

Forward secrecy: forward secrecy means that the group key must be updated when a member leaves the group. This is to prevent an old member from deciphering current and future group communication. Assume U_5 in Figure 3 leaves the group, K_0, K_2 and SK must be changed to K_0(new), K_2(new) and SK(new). The key server generates SK(new) and broadcasts the following message: $\{SK(\text{new})\}K_1$, $\{SK(\text{new})\}\bar{K_0}$, $\{SK(\text{new})\}\bar{K_1}$, $\{SK(\text{new})\}\bar{K_2}$. Upon receiving the message, the remaining users can obtain SK(new) by decrypting one part of the message. Then, the new keys: K_0(new) and K_2(new) are calculated using one-way functions of the old keys and the new group key, i.e. $K_0(\text{new}){=}f(SK(\text{new}),K_0)$, $K_2(\text{new}){=}f(SK(\text{new}), K_2)$. Since U_5 does not know the K_1, $\bar{K_0}$, $\bar{K_1}$ and $\bar{K_2}$, therefore it can't obtain or calculate K_0(new), K_2(new) and SK(new). This means that it cannot access future group communications.

Collusion attacks: collusion attacks means that evicted members are able to work together and share their individual piece of information to regain access to the group key. In the proposed protocol, as in KM-BFM protocol, two evicted members can co-operate together to regain access to the group key. For the example given in Section 2.3, U_3 and U_{12} can work together to get the new session key. In this example, the group manager encrypts the new session key using K_0, K_2, $\bar{K_0}\bar{K_1}$, $\bar{K_0}\bar{K_2}$ and $K_1\bar{K_2}$. It has to be noted that U_3 knows $\bar{K_0}$ and K_1. On the other hand, U_{12} knows $\bar{K_1}$ and $\bar{K_2}$. Therefore, they could co-operate together to calculate $\bar{K_0}\bar{K_1}$, $\bar{K_0}\bar{K_2}$ and $K_1\bar{K_2}$ and get the new session key. In order to overcome this problem, we suggest the following steps:

Step 1: Construct a table having all members as columns and all auxiliary key patterns as rows.

Step 2: Put 1 in cell (i, j) to indicate that U_j knows at least one key in pattern P_i.

Step 3: When member(s) U_j leaves the group, delete all row(s) that have 1 in column(s) j, then delete all column(s)j.

Step 4: Select a row which contains maximum number of 1's (cover maximum members). In the case where two rows have the same number of 1's, choose the row which contains a user with the least column index. Then, put this Pi to be the subset T_k, where the initial value of k is 1. If there isn't any 1's in the remaining rows, then go to Step 7.

Step 5: Remove all members U_j that have 1 in row P_i. Then, remove row P_i.

Step 6: If $\cup T_i \neq U_r$, increase k by one and go to Step 4.

Step 7: For the remaining users, the key server can use a shared key between it and each user to encrypt the new group key. Table 7 represents Steps 1 and 2.

Applying Step 3 after removing both U_3 and U_{12} will lead to Table 8. Next, applying Step 4 leads to Table 9. Therefore, the first subset is $T_1 = P_1 =\{U_1,U_2,U_5,U_6\}$. After removing users contained in P_1, we will obtain Table 10.

Applying Step 4 again, we will obtain the second subset $T_2 = P_5 = \{U_7, U_9, U11\}$. After removing users contained in P_5, we will obtain Table 11.

Therefore $T_3 = K_{S,U_4}$, $T_4 = K_{S,U_8}$ and $T_5 = K_{S,U_{10}}$, where K_{S,U_i} is the symmetric key shared between the key server and U_i. Finally, union of T_1, T_2, T_3, T_4 and T_5 will cover all the remaining group members. This modification will overcome the collusion attack problem with higher storage requirements at the group manager. It has to be noted that in LKH protocol, the server has to store $2m$ keys, where m represents the number of group members. While after making the above mentioned modification, the server in the proposed protocol has to store at most $m+\log_2(m)$ keys. Therefore, the proposed protocol still has a lower storage requirements compared to the LKH protocol. On the other hand, KM-BFM protocol also suffers from the collusion attack problem. Therefore, in order to overcome this problem, the storage requirements at the server must also increase. In the next section, a comparison of the proposed protocol with other centralized key distribution protocols is detailed.

3 Comparison of Centralized Key Distribution Protocols

We, first, compare centralized approaches: LKH, OFT and KM-BFM protocols. Since, KM-BFM protocol has the lowest communication overhead and it has the lowest storage requirements as will be shown. Therefore, a comparison between KM-BFM and the proposed protocol is detailed. In order to conduct the comparison of centralized approaches, the following general assumptions are considered:

- The total number of group members equals to Ng members.
- The tree height is h, where $h = \log_2(Ng)$.
- The time of encryption operation equals to the time of decryption operation.
- The one-way function can be carried out using a hash algorithm like SHA-512.
- The time required to make an encryption operation is twice that of calculating a hash operation.

The comparison will be undertaken according to the following criteria:

- Satisfying the forward and backward secrecies requirements.
- Eliminating the problem of collusion attacks.
- The number of encryption operations needed to be performed by Security Manager (SM), the requesting user and non-requesting users in case of a member leaves or joins.
- The length of re-key message needed to update the group key after a member leave/join.
- Storage required at SM and users.

Table 12 shows the comparison between LKH, OFT and KM-BFM protocols. From the table the following facts could be deduced:

- Both OFT and KM-BFM protocols have lower communication and computation overheads compared to LKH protocol.

Table 7: All key patterns and each corresponding user

	U_1	U_2	U_3	U_4	U_5	U_6	U_7	U_8	U_9	U_{10}	U_{11}	U_{12}
P_1	1	1			1	1						
P_2			1	1			1	1				
P_3	1		1						1	1		
P_4		1		1							1	1
P_5					1		1		1		1	
P_6						1		1		1		1
P_7	1	1	1		1	1			1	1		
P_8	1	1		1	1	1					1	1
P_9	1		1	1			1	1	1	1		
P_{10}		1	1	1			1	1			1	1
P_{11}	1	1			1	1	1		1		1	
P_{12}	1	1			1	1		1		1		1
P_{13}			1	1	1		1	1	1		1	
P_{14}			1	1		1	1	1		1		1
P_{15}	1		1		1		1		1	1	1	
P_{16}	1		1			1		1	1	1		1
P_{17}		1		1	1		1		1		1	1
P_{18}		1		1		1		1		1	1	1

Table 8: Step 3

	U_1	U_2	U_4	U_5	U_6	U_7	U_8	U_9	U_{10}	U_{11}
P_1	1	1		1	1					
P_5				1		1		1		1
P_{11}				1						

Table 9: Applying Step 4

	U_1	U_2	U_4	U_5	U_6	U_7	U_8	U_9	U_{10}	U_{11}
P_1	1	1		1	1					
P_5				1		1		1		1
P_{11}				1						

Table 10: Removing users contained in P_1

	U_4	U_7	U_8	U_9	U_{10}	U_{11}
P_5		1		1		1
P_{11}						

Table 11: Removing users contained in P_5

	U_4	U_8	U_{10}
P_{11}			

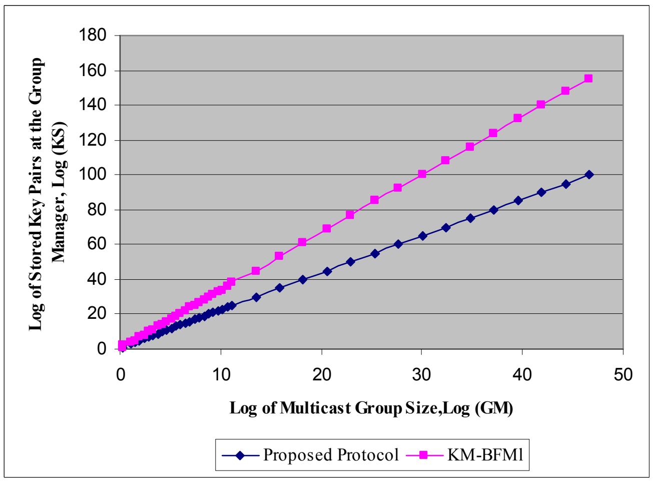

Figure 5: Key storage comparison at group manager between the proposed protocol and KM-BFM protocol

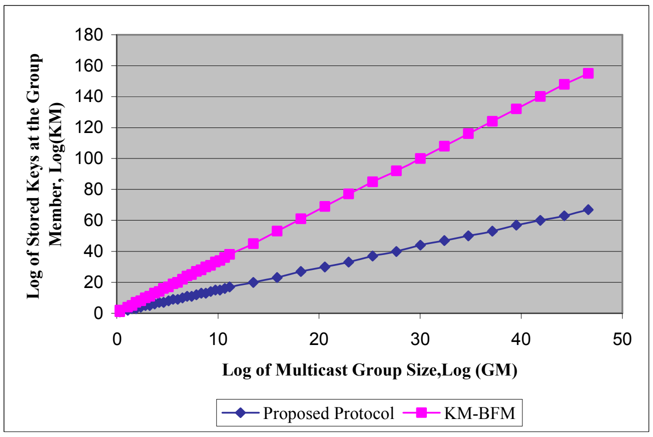

Figure 6: Key storage comparison at group member between the proposed protocol and KM-BFM protocol

- Both OFT and KM-BFM protocols have the same computation overhead. The main advantage of OFT compared to KM-BFM is that it can resist collusion attacks. On the other hand, the main advantage of KM-BFM is that it has a lower communication overhead and it requires lower storage requirements at SM.

In the following, a comparison of the proposed protocol with KM-BFM protocol is undertaken. The comparison is made according to:

- Storage requirements at both group manager and group members.

- Number of updates in case of a single leave or worst case of multiple leaves.

In order to deduce the comparison, the following parameters are assumed:

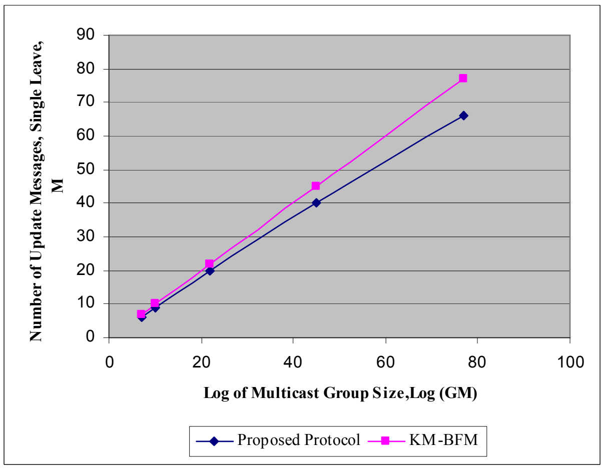

Figure 7: Update messages comparison between the proposed protocol KM-BFM protocol, single leave

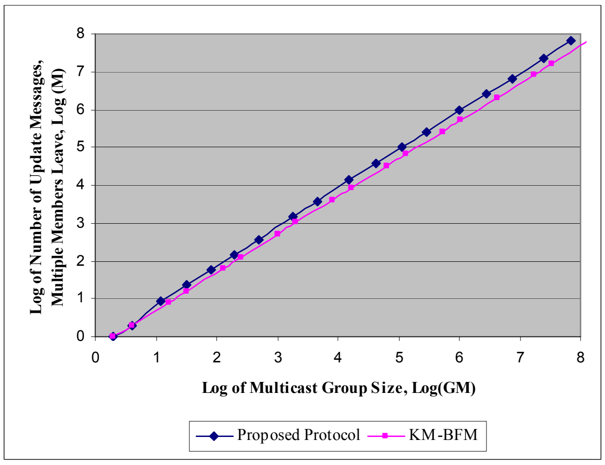

Figure 8: Update messages comparison between the proposed protocol KM-BFM protocol, multiple leave

Table 12: Comparison of the proposed protocol with KM-BFM protocol

	LKH	**OFT**	**KM-BFM**
Forward secrecy	Yes	Yes	Yes
Backward secrecy	Yes	Yes	Yes
Secure against collusion attacks	Yes	Yes	No
Member join: SM encryption operations	$2h$ keys	$\frac{3h}{2}+2$ keys	$\frac{3h}{2}+2$ keys
Requesting member's encryption operations	h keys	h keys	h keys
Non- requesting member's encryption operations	2 key	$\frac{h}{4}+1$ keys	$\frac{h}{4}+1$ keys
Re-key message length	$2h+1$ keys	$2h+1$ keys	$h+2$ keys
Member leave: SM encryption Operations	$h(h+1)/2$ keys	$\frac{3h}{2}$keys	$\frac{3h}{2}$keys
Non- requesting member's encryption operations	2	$\frac{h}{4}+1$ keys	$\frac{h}{4}+1$ keys
Re-key message length	$h(h+1)/2$ keys	h keys	h keys
Storage requirements: At SM	$2Ng-1$ keys	$2Ng$ -1 keys	$2h$ keys
At each user	$h+1$ keys	$h+1$ keys	$h+1$ keys

- GM = total number of group members.
- KS = total number of key pairs at the group manager.
- KM = total number of keys at each group member.
- M = total number of update messages.
- m = number of key pairs hold by the group manager in KM-BFM protocol.
- n = number of key pairs hold by the group manager in the proposed protocol.

Figures 5 and 6 represent the storage requirements at both the group manager (KS) and group members (KM) against the total number of group members (GM). While, Figure 5 shows the shared key pairs at the group manager, Figure 6 illustrates the stored keys at each group member. The total number of group members 'GM' is calculated using Equation (1) for the proposed protocol and equals to 2^m members for KM-BFM protocol. The number of stored key pairs at the group manager equals 'n' in the proposed protocol, while in KM-BFM protocol, the stored key pairs at the group manager equals 'm'. The storage at each member in the proposed protocol equals 'y' keys as shown in Table 1, while in KM-BFM protocol, the number of stored keys at each member is 'm'. Figure 5 On the other hand, Figure 7 and Figure 8 represent the number of update messages (M) in case of a single leave or in the worst case against GM. While, Figure 7 shows the number of update messages in case of a single leave, Figure 8 illustrates the number of update messages in the worst case. For the proposed protocol, in case of a single leave, the number of update messages equals to $(2n--y)$. On the other hand, for KM-BFM protocol, the number of update messages is m. Concerning the worst case in case of multiple members leave, for the proposed protocol, the total number of update messages is given by Equation (4). On the other hand, for KM-BFM protocol, the total number of update messages equals 2^{m-1} [4]. Figure 5, Figure 6 and Figure 8 are based on a $\log_{10}$ scale for the two axes. From Figure 5 to Figure 8, the following facts could be deduced:

- The proposed protocol has a lower storage at both the group manager and group members as shown in Figure 5 and Figure 6. Therefore, this satisfies the requirements of multicast key distribution protocol as mentioned in [14].
- For the case of a single member leave, the proposed protocol has a number of update messages that is less than that of KM-BFM protocol, which leads to a better bandwidth utilization as shown in Figure 7.
- Figure 8 shows that the proposed protocol has a comparable number of update messages as in KM-BFM protocol in the worst case.

In conclusion, the proposed protocol achieves lower storage requirements at both the group manager and the group members for the same number of group members. On the other hand, it has a lower communication overhead in case of a single member leave and a comparable communication overhead in case of multiple leaves.

4 Conclusions

In the present paper, the problem of securing multicast communication is discussed. Many solutions have been proposed to solve the problem of distributing a symmetric key between the whole group members. A nave solution is to include a group manager that shares a different symmetric key with each group member. Any change in the group membership will lead to the change of the group key. The group manager generates a new group key and encrypts it using the symmetric keys shared between it and the group members. This solution is not scalable and its computation and communication overheads increase linearly with the number of group members. Other protocols achieve a lower communication and computation overheads, which are of $O(\log N)$, where N represents the group member. These protocols are based on constructing a tree of keys and known as LKH protocols. KM-BFM protocol is considered an enhancement to LKH protocols, it achieves lower communication and computation overheads, which are half those of LKH protocols. In the present paper, a scalable protocol for securing multicast communication is proposed. The proposed protocol is based on KM-BFM protocol. Instead of using one tree as in KM-BFM, the members are divided into a number of subgroup trees. A comparison between the proposed protocol and KM-BFM protocol is undertaken according to storage requirements at both group manager and group members and the number of updates in case of a single leave or worst case multiple leaves. The comparison shows that the proposed protocol achieves lower storage requirements at both the group manager and the group members for the same number of group members. On the other hand, it has a lower communication overhead in case of a single member leave and a comparable communication overhead in case of multiple leaves.

References

[1] A. Ballardie, *Scalable Multicast Key Distribution*, RFC 1949, 1996.

[2] K. Becker, and U. Wille, "A communication complexity of group key distribution," *Proceedings of the 5th ACM Conference on Computer and Communications Security*, pp. 1-6, San Francisco, California, Nov. 1998.

[3] R. Canetti, J. Garary, G. Itkis, D. Micciancio, M. Naor and B. Pinkas,"Multicast security: A taxonomy and some efficient Constrictions," *Proceedings of the IEEE INFOCOM*, vol. 2, pp. 708-716, New York, Mar. 1999.

[4] I. Chang, R. Engel, D. Kandlur, D. Pendarakis, and D. Saha,"Key management for secure internet multicast using Boolean function minimization techniques," *Proceedings of the IEEE INFOCOM*, vol. 2, pp. 689-698, New York, Mar. 1999.

[5] L. Dondeti, S. Mukherjee and A. Samal, "Scalable secure one-to-many group communication using dual encryption," *IComputer and Communication*, vol. 23, no. 17, pp. 1681-1701, Nov. 1999.

[6] P. Golle, and N. Modadugu,"Streamed authentication in the presence of random packet loss," *Proceedings of the ISOC Network and Distributed System Security Symposium*, pp. 13-22, California, USA, Feb. 2001.

[7] M. Goodrich, and R. Tamassia, *Algorithm Design: Foundation, Analysis and Internet Examples*, John Wiley and Sons Inc., 2002.

[8] Y. Kim, A. Perrig, and G. Tsudik, "Simple and fault-tolerant key agreement for dynamic collaborative groups," *Proceeding of the 7th ACM Conference in Computer and Communication Security*, pp. 235-244, Athens Greece, Nov. 2000.

[9] E. Kreyszig, *Advanced Engineering Mathematics, John Wiley - Sons Inc.*, Seventh Edition, 1993.

[10] P. Kruus, and J. Macker, "Techniques and issues in multicast seecurity," *Proceedings of IEEE MILCOM '98*, Oct. 1998.

[11] D. McGrew, and A. Sherman, *Key Establishment in Large Dynamic Groups Using One-Way Function Trees*, Technical Report No. 0755, TIS Labs at Network Associates, Inc., Glenwood, MD, May 1998.

[12] S. Mittra, "Iolus: A framework for scalable secure multicasting," *Proceedings of the ACM SIGCOMM*, vol. 27, no. 4, pp. 277-288, New York, Sep. 1997.

[13] A. Perrig, R. Canetti, D. Tygar, and D. Song,"The TESLA broadcast authentication protocol," *RSA CryptoBytes*, vol. 5, pp. 2-13, 2002.

[14] S. Rafaeli, and D. Hutchison,"A survey of Key Management for Secure Group Communication," *ACM Computing Surveys (CSUR)*, vol. 35, no. 3, pp. 309-329, Sep. 2003.

[15] O. Rodeh, K. Birman, and D. Dolev, "Optimized group re-key for group communication systems," *Network and Distributed System Security*, pp. 37-48, San Diego, California., Feb. 2000.

[16] S. Setia, S. Zhu, and S. Jajodia,"Kronos: A scalable group re-keying approach for secure multicast," *Proceeding of the IEEE Symposium on Security and Praivcy*, pp. 215-228, Oakland, California, May 2000.

[17] S. S. Skiena, *The Algorithm Design Manual*, Springer-Verlag, New York, 1997.

[18] G. Thomas, and R. Finney, *Calculus and Analytic Geometry*, Addison-Wesley Publishing Company Inc., Eighth Edition, 1992.

[19] X. Verians, and J. M. Boucqueau, *Next Generation Conditional Access System for Satellite Broadcasting*, Final Report, Contract no. 16696/02/NL/US, Published on Jan. 12, 2004.

[20] E. Weisstein, *Digamma Function*, On Line Article accessed from Internet in Oct. 2004. (http:// mathworld.wolfram.com/DigammaFunction.html)

[21] C. Wong, M. Gouda, and S. Lam, "secure group communications using key graphs," *Proceedings of ACM SIGCOMM*, pp. 68-79, Vancouver, British Columbia, September 1998.

Mohamed Rasslan has his M. Sc in Computer and System Engineering and he received his B. Sc. from Faculty of Engineering, Cairo University.

Yasser Dakroury is a Professor of Computer Engineering, Computer and Systems Engineering Department, Faculty of Engineering, Ain Shams University, Cairo - Egypt. Currently, he is the Managing Director of the Egyptian E-Learning University (EELU) project. He has many research and consultation activities in the following areas: Real Time Systems, Computer Networks, Networks and Internet Security, Cryptography Engineering, Software Engineering, E-Commerce and E-Leaning. He has supervised + 20 Ph.D. and M.Sc. theses. He has + 50 published papers in international conferences and journals.

Heba K. Aslan is an Associate Professor at the Electronics Research Institute, Cairo- Egypt. She received her B. Sc., M. Sc. And Ph. D. from Faculty of Engineering, Cairo University in 1990, 1994 and 1998 respectively. Her research interests include: Key Distribution Protocols, Authentication Protocols, Logical Analysis of Protocols and Intrusion Detection Protocols.