International Scholarly Research Network
ISRN Communications and Networking
Volume 2012, Article ID 383218, 11 pages
doi:10.5402/2012/383218

Research Article

Key Management Schemes for Multilayer and Multiple Simultaneous Secure Group Communication

R. Aparna[1] and B. B. Amberker[2]

[1] *Department of Computer Science and Engineering, Siddaganga Institute of Technology, Tumkur, Karnataka 572103, India*
[2] *Department of Computer Science and Engineering, National Institute of Technology, Warangal, Andhra Pradesh 506004, India*

Correspondence should be addressed to R. Aparna, r_aparna27@yahoo.com

Received 22 March 2012; Accepted 18 April 2012

Academic Editors: G. Horng and H.-L. Wei

Many emerging applications are based on group communication model and many group communications like multimedia distribution and military applications require a security infrastructure that provides multiple levels of access control for group members. The group members are divided into a number of subgroups and placed at different privilege levels based on certain criteria. A member at higher level must be capable of accessing communication in its own level as well as its descendant lower levels but not vice versa. In this paper we propose a key management scheme for this multilayer group communication. We achieve substantial reduction in storage and encryption cost compared to the scheme proposed by Dexter et al. We also address periodic group rekeying. Applications like scientific discussion and project management may lead to a scenario in which it is necessary to set up multiple secure groups simultaneously, and few members may be part of several secure groups. Managing group keys for simultaneous secure groups is critical. In this paper we propose a novel key management scheme for multiple simultaneous groups.

1. Introduction

Many emerging applications like secure audio and visual broadcasts, pay-per-view, scientific discussion, and teleconferencing are based on group communication model. Several users participate in these applications, and multicast communication is an efficient means of distributing data to a large group of participants [1–3] since it reduces the demands on network and bandwidth resources. But, the communication among these participants must be carried out confidentially. Thus, a common key known as *group key* or *secret key* must be established with all the users in the group, so that any group member can encrypt the message using this key, and all others can decrypt the message using the same key. The group, being dynamic in nature, allows member join and leave events. Efficiently managing group key for large, dynamically changing groups is a difficult problem. Every time when a new member joins the group, the group key must be changed in order to provide *backward access control* (i.e., new members should not be able to access past communication). Similarly, when a user leaves the group, the group key must be changed so that leaving member cannot have access to future communication that takes place between remaining group members, known as *forward access control*. This group key updating process is referred to as *rekeying*.

Rekeying process involves changing the group key whenever there is a membership change and distributing it among the members of the group in a secure manner. To communicate changed key among group members securely, rekey messages are constructed, encrypted, and multicast to the group. The overhead involved in rekey operation, that is, key updation, number of encryptions performed, and communication cost must be minimum and should be independent of the group size, which improves scalability.

Several secure group key management techniques have been proposed to support scalable secure multicasting [4–10]. In a typical multicast key management scheme, there is a trusted third party, known as Key Distribution Center (KDC). This single trusted centralized entity is responsible for generating and distributing keys securely to the group members. Among the schemes which involve KDC [7, 11–15], the scheme proposed by Wong et al. in [7] is efficient and is widely used since it improves scalability. The scheme uses a hierarchical tree structure in which users are maintained at the leaf level, and every user is assigned with keys along the path of its location till the root. Besides group key, the KDC shares auxiliary keys that are used solely for the purpose of updating the group key and other auxiliary keys. In addition, every user shares a private key that is known by itself and the KDC. These schemes are referred to as *key-based* schemes.

Hierarchical tree structure is also used in Centralized Key Management with Secret Sharing (CKMSS) [16, 17]. In this scheme, KDC considers a t degree (t is a nonzero positive integer) polynomial with the constant term of the polynomial being the secret key. It computes t distinct shares known as *prepositioned information* and stores them at the users. To compute the group key, $(t + 1)$ shares of the polynomial are required, and this $(t + 1)$th share is sent as an *activating share* by the KDC. Once the group key is computed, it is used until a member joins or leaves the group. For every membership change (join/leave), to perform rekey operation, KDC multicasts an activating share to enable the members to compute new group key. These schemes are called *share-based schemes*.

Both the key-based and share-based schemes discussed above are designed for managing keys for a group of users enjoying same privilege and are not suitable for handling multilayer and multiple SGC scenario. But, for certain applications, it is necessary to have multilayer group communication scenario where members in the system have different privileges. In some applications a member u in the system may be part of several groups. In this paper, we address the above two cases of Secure Group Communication (SGC) and propose key management schemes.

We organize the paper as follows: Section 2 focuses on the applications of multilayer SGC and highlights the schemes proposed to address such scenario in detail. Section 3 concentrates on our scheme to manage multilayer hierarchy. We discuss initial key computation, rekeying during join/leave operation, periodic rekeying. In section 4 we compare the performance of our scheme with Dexter et al. scheme. Section 5 deals with setting up multiple groups, initial key computation, and rekeying. Section 6 presents authentication to multiple SGC, and we conclude the paper in Section 7.

2. Applications of Multilayer SGC

(i) In multimedia applications, we can consider two categories of receivers: high-definition television (HDTV) and traditional television. Users with HDTV receivers can form one subgroup and others with traditional television receiver can form another subgroup. Users with traditional television receivers can receive the normal format, while the users with HDTV receiver can receive both the normal format and the extra information needed to achieve HDTV resolution. Thus, there are two layers, group with HDTV receiver forms higher layer subgroup and the one with traditional television receiver forms lower layer subgroup. This application requires a multilayer SGC scenario.

(ii) In multicast scalable video service, the video is encoded into 3 quality levels: basic quality level, medium quality level, and best quality level. Here, the users can be classified into 3 different layers based on the quality of the video they purchase: base layer (BL), enhancement layer 1 (EL1), and enhancement layer 2 (EL2). The users purchasing the basic video quality level belong to BL group, users purchasing the medium quality level belong to both BL and EL1 groups, whereas the users purchasing the best quality level belong to all the three, that is, BL, EL1, and EL2 groups. Thus, users with access to higher-quality video service must also have access to lower-quality ones.

(iii) Military troop contains different categories like Captains, Lieutenants, Sergeants, Corporals, Soldiers, and so forth, and this requires a hierarchical group communication model. Captains are at the highest layer, Lieutenants at the second higher level layer, Sergeants at a layer below Lieutenants, Corporals at the next lower layer, and Soldiers must be at the lowest layer as considered in [18]. Soldiers should be able to communicate only with other Soldiers (peer members), whereas Sergeants can communicate with other Sergeants as well as with Corporals and Soldiers. Similarly Captains should have access to all the communications that take place between different classes.

(iv) In project management, a single project is divided into multiple modules, and set of users are made to design a particular module. Users involved in handling one module form one secure group. This may lead to a scenario in which it is necessary to set up multiple secure groups simultaneously.

To manage the above type of scenarios, a naive solution is to extend key-based and share-based tree structure, by using independent trees for each layer. But, this is inefficient and does not scale well when there are many layers. Hence, there is a need to have a multigroup key management scheme that exploits the overlap in the memberships of different layers. Two key management schemes have been developed to provide hierarchical access control. The scheme proposed in [19, 20] is key-based, where each layer has its own session key and whenever there is a membership change in any layer, corresponding session key is changed and securely transmitted to appropriate group members. The scheme proposed in [21] is share based. For each layer, a polynomial of degree t is considered, and t distinct shares of this polynomial are stored at the members of that layer (prepositioned information) and KDC sends $(t + 1)$th share as an activating share so that members can compute group key for that layer. Whenever there is a membership change in any layer, KDC just sends a different activating share to the members of that layer so that they can compute a new group key for that layer.

In [18], a military application is considered to illustrate multilayer secure group communication. Military officers belonging to different categories (Captains, Lieutenants, Sergeants, Corporals, Soldiers, etc.) are divided into subgroups and are hierarchically placed one above the other. Higher

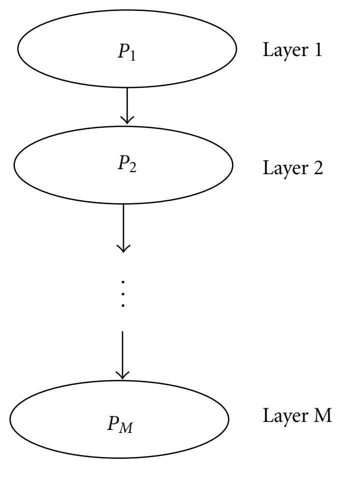

Figure 1: Subgroups arranged into different privilege layers.

layer officials can have access to the communication between its descendant lower-layer subgroups. To provide this feature, it uses a one-way hash function $H(\cdot)$ to compute a chain of keys. The main idea of using $H(\cdot)$ function is to relate layers' keys in such a way, that is, knowing a key of its own layer, a member can compute keys of lower layers. Thus, Captains are given with random key K, for the Lieutenants $H(K)$ is given, for Sergeants $H(H(K)) = H^2(K)$ is sent, Corporals are assigned with $H^3(K)$, and Soldiers with $H^4(K)$. Captains can access the communication between the Sergeants, by computing the key $H^2(K)$.

In [19, 20], Sun and Liu used tree-based hierarchical approach to handle broadcasting multimedia applications in different layers to different groups of users.

To the best of our knowledge, only SGC within a single group is addressed in the literature. A SGC among multiple groups is not addressed in the literature. In this paper we address key management schemes for multilayer and multiple groups and our schemes can be used for the applications explained above.

2.1. Multilayer Secure Group Communication. Multilayer key management scheme proposed in [19, 20] uses the following model: a set of users $U = \{u_1, u_2, \ldots, u_N\}$ is partitioned into M subsets (subgroups) $P_1, P_2, \ldots, P_M$ such that members of P_i and can communicate with each other. However, members of P_i can communicate with members of P_j, $i > j$, but not vice versa, $1 \leq i, j \leq M$. We say that the members of subgroup P_i are at layer i and members of subgroup P_i belong to subgroup P_j, $i > j$. Members of subgroup P_i overlap with the members of subgroup P_j, $i > j$. Figure 1 illustrates the arrangement of different subgroups P_i in layered approach, $i = 1, 2, \ldots, M$.

To manage keys for multiple layers in traditional hierarchical tree-based key management scheme, a separate key tree is constructed for each layer. Although it is easy to implement using independent trees, a substantial overhead is introduced in managing the keys due to the overlapping membership in different layers.

In order to manage the keys of all the subgroups, independent trees are integrated into one key graph in [19, 20], and it uses a key-based scheme to manage the keys. In [21], integrated key graph as in [19, 20] is used, but for key management it uses share-based scheme. The key management scheme proposed in [21] is explained as follows:

(1) KDC fixes the security parameter t,

(2) KDC constructs a Logical Key Tree (LKT) as in [7] for the subgroup P_i at layer i. For a secure group with S_i users, there are at most $2S_i - 1$ nodes in LKT and the height of the LKT, is $\lceil \log_2 S_i \rceil$,

(3) for each node in LKT of subgroup P_i, KDC selects randomly $t - 1$ number of distinct points (x^i_{kj}, y^i_{kj}) in GF(p) (where GF refers to Galois Field) called *prepositioned shares*, $i \in \{1, 2, \ldots, M\}$, $j = 1, 2, \ldots, t - 1$, $k = 1, 2, \ldots, 2S_i - 1$. To each user $u \in P_i$, KDC sends *securely* $(t - 1)\log_2 S_i$ shares pertaining to the shares for the nodes along the path from leaf node u till root,

(4) KDC selects another point (x_t, y_t) called activating share (AS) and broadcasts it to the members of all the layers in the system,

(5) a member of the subgroup P_i constructs $\log_2 S_i$ number of polynomials of degree $(t - 1)$ using the corresponding shares it has received from KDC and AS, $i = 1, 2, \ldots, M$ and evaluates each polynomial at 0 to get the keys, and

(6) KDC also constructs polynomial of degree $(t - 1)$ for each node k in the LKT of P_i using the $t - 1$ points (x^i_{kj}, y^i_{kj}) and AS and evaluates at 0 to get the corresponding key for that node.

In this scheme, each key is computed by constructing a $t - 1$ degree polynomial using $t - 1$ different prepositioned shares and a common activating share. In this scheme, each user u_i is required to store prepositioned shares of the nodes from leaf to the root.

3. Proposed Key Management Scheme for Multilayer SGC

We propose to use the key graph structure as in [19, 20]. We construct individual key trees for different layers and then integrate them. We use the same model that is explained for Dexter et al. scheme and try to reduce the amount of storage required at both KDC and users [22]. For auxiliary keys, we use random elements, and we compute the group keys as described below.

KDC fixes the security parameter t, computes, and distributes the keys and shares as follows:

(1) KDC selects randomly $t - 1$ number of points (x_i, y_i) in GF(p) called *prepositioned shares*, $i = 1, 2, \ldots, t - 1$ and an *activating share* (AS) and sends *securely* to the members of all the subgroups $P_1, P_2, \ldots, P_M$,

(2) KDC constructs LKT for the subgroup P_1 at layer 1 with the *subgroup key* G_1. The key G_1 is obtained by

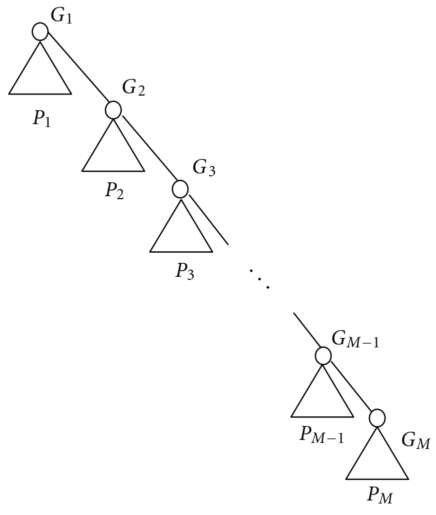

Figure 2: Hierarchical key tree structure for multilayer secure group communication with M layers.

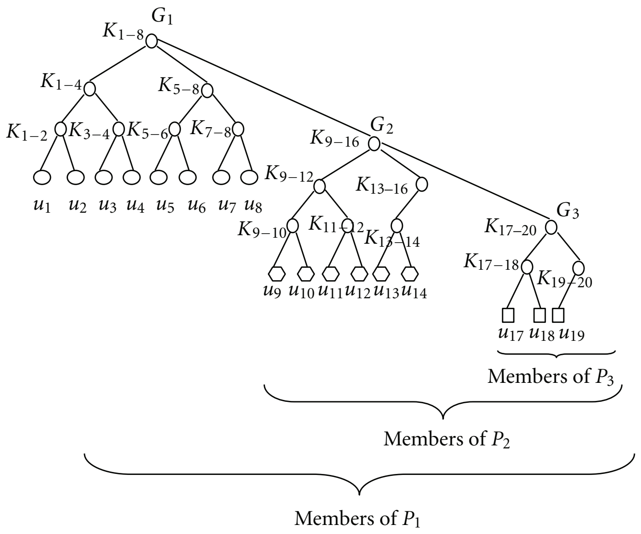
G_1
K_{1-8}
K_{1-4}
K_{5-8}
K_{1-2} K_{3-4} K_{5-6} K_{7-8}
u_1 u_2 u_3 u_4 u_5 u_6 u_7 u_8
G_2
K_{9-16}
K_{9-12}
K_{13-16}
K_{9-10} K_{11-12} K_{13-14}
u_9 u_{10} u_{11} u_{12} u_{13} u_{14}
G_3
K_{17-20}
K_{17-18} K_{19-20}
u_{17} u_{18} u_{19}
Members of P_3
Members of P_2
Members of P_1

Figure 3: An example integrated key graph with auxiliary keys and group keys.

constructing a polynomial $P(x)$ of degree $t-1$ using the $t-1$ points (x_i, y_i) and AS and evaluating at 0 to get $G_1 = P_1(0)$. KDC sends *secretly* to each user u of P_1, all the auxiliary keys along the path of LKT from leaf u to the root,

(3) KDC selects group share (xg_j, yg_j) for the subgroup P_j and sends secretly to the members of the subgroup $P_j, P_{j+1}, \ldots, P_M$, $j = 2, 3, \ldots, M$, and

(4) KDC constructs LKT for the subgroup P_j at layer j with the subgroup key G_j. The key G_j is obtained by constructing a polynomial $P_j(x)$ of degree $t+j-2$ using $t-1$ prepositioned shares, AS, and $j-1$ group shares (xg_l, yg_l), $l = 2, 3, \ldots, j$ and $j = 2, 3, \ldots, M$. It evaluates the polynomial P_j at 0 to get $G_j = P_j(0)$. To each user u of P_j, KDC sends *secretly* all the auxiliary keys along the path of LKT from leaf u to root.

Figure 2 shows the hierarchical key tree structure with M layers.

Figure 3 shows an example integrated key graph for three layers in which three independent groups are integrated to form a three-layer hierarchy. In Figure 3, G_1, G_2, and G_3 represent the roots of the subgroups P_1, P_2, and P_3, respectively.

We are addressing the following events:

(1) a new member joins the service,

(2) a member leaves the service, and

(3) a member moves from one service layer to another service layer.

3.1. Member Join Event. When a new member, u_{new}, joins any layer i, $i = 1, \ldots, M$, keys along the path from the joining point till the root must be changed and conveyed to corresponding users. Instead of KDC changing the keys or sending a new activating share, we allow the members of layer i themselves to compute the new group key and auxiliary keys on their own by applying one-way hash function to the corresponding previous keys. KDC also applies one-way hash function to the previous keys on the path from the joining point till the root. Members at layers $i+1$ to M change the group key G_i by applying one-way hash function to previous group key G_i. For the new user, u_{new}, KDC sends $t-1$ prepositioned shares (x_i, y_i), auxiliary keys of P_i along the path to where u_{new} is inserted, and group key G_i and i group shares (xg_i, yg_i) by encrypting with the private key of u_{new}, $i = 1, 2, \ldots, M$.

3.2. Member Leave Event. If a member at layer i leaves, $i = 1, 2, \ldots, M$, the following keys have to be changed:

(1) keys along the path from the leaving position till root,

(2) subgroup key G_i, and

(3) subgroup keys $G_1, G_2, \ldots, G_{i-1}$ of layers from 1 to $i-1$.

KDC generates keys along the path and sends them securely to the required members of the group. KDC generates and sends a new activating share securely to the members of the subgroups $P_1, P_2, \ldots, P_i$. Users of group P_j construct the polynomial $P_j(x)$ using prepositioned shares, group share, and new activating share and compute the group key G_j by evaluating the polynomial $P_j(x)$ at 0, $j = 1, 2, \ldots, i$.

We illustrate this with the following example. From Figure 3, if member u_8 leaves the group, the following messages are constructed and sent to users ($\{K_i\}_{K_j}$ indicates key K_i encrypted with key K_j and AS denotes activating share)

$$\text{KDC} \rightarrow \{u_1 \text{ to } u_4\} : \{\text{AS}\}_{K_{1-4}}$$
$$\text{KDC} \rightarrow \{u_5, u_6\} : \{\text{AS}\}_{K'_{5-8}}, \{K'_{5-8}\}_{K_{5-6}}$$
$$\text{KDC} \rightarrow u_7 : \{\text{AS}\}_{K'_{5-8}}, \{K'_{5-8}\}_{K'_{7-8}}, \{K'_{7-8}\}_{K_7}$$
$$\text{KDC} \rightarrow \{u_9 \text{ to } u_{14}\} : \{\text{AS}\}_{G_2}$$
$$\text{KDC} \rightarrow \{u_{17} \text{ to } u_{19}\} : \{\text{AS}\}_{G_3}.$$

3.3. Member Moving from One Service Layer to Another. Here we encounter two cases.

Case 1. If a member moves from layer i to its higher layer j $(i < j)$, then it must be provided with extra group share/s meant for layers from $(i + 1)$ to j. To provide backward confidentiality for the messages communicated in the subgroups from $i + 1$ to j, group share (xg_k, yg_k) of the group P_k is changed, $k = i + 1, i + 2, \ldots, j$. KDC generates group share (xg_k, yg_k) and encrypts with the previous group key G_k and sends to members of the group P_k, $k = i + 1, i + 2, \ldots, j$.

Case 2. If a member moves from layer i to its descendant layer j, $(i < j)$, then group shares of layers from $(i + 1)$ to j must be changed. In layer j, auxiliary keys possessed by the moving member must be changed. To convey new group share (xg_k, yg_k) to the members of subgroup P_k, it is encrypted with the auxiliary keys at level 1 of LKT of subgroup P_k, $k = i + 1, i + 2, \ldots, j$.

Example 1. In Figure 3, if a member u_{12} moves from layer 2 to layer 3, it is inserted as sibling of member u_{19}. The group share (xg_3, yg_3) must be changed to provide backward access control. The following messages are constructed and sent to users:

$$\text{KDC} \rightarrow \{u_{17}, u_{18}\} : \{(xg_3, yg_3)\}_{G_3}, \text{AS}$$

$$\text{KDC} \rightarrow u_{19} : \{(xg_3, yg_3)\}_{G_3}, \{K'_{19-20}\}_{K_{19}}, \text{AS}$$

$$\text{KDC} \rightarrow u_{12} : \{(xg_3, yg_3), K'_{19-20}\}_{K_{12}}, \text{AS}$$

$$\text{KDC} \rightarrow \{u_9, u_{10}\} : \{K'_{9-16}\}_{K'_{9-12}}, \{K'_{9-12}\}_{K_{9-10}}$$

$$\text{KDC} \rightarrow u_{11} : \{K'_{9-16}\}_{K'_{9-12}}, [K'_{9-12}]_{K'_{11-12}}, \{K'_{11-12}\}_{K_{11}}$$

$$\text{KDC} \rightarrow \{u_{13}, u_{14}\} : \{K'_{9-16}\}_{K_{13-16}}.$$

3.4. Periodic Rekeying. If the content has very high value, even though there is no membership change, group key must be changed for all the layers periodically. This leaves the attacker with very less time to attack on the current key values. To achieve this periodic rekeying, we fix a rekey period/interval. After the expiry of each rekey interval, rekeying process is initiated. For periodic rekeying, we propose two methods.

Method 1. (i) KDC sends an activating share by encrypting it with layer 1 group key, G_1.

(ii) Since all the users $u_1, u_2, \ldots, u_N$ in the system belong to subgroup P_1, they know the subgroup key G_1 and can decrypt the activating share.

(iii) Users of subgroup P_i compute new subgroup keys $G_1, G_2, \ldots, G_i$ using new activating share, prepositioned shares and group share (xg_i, yg_i).

Method 2. The users compute the new group key after every rekey interval by applying a one-way hash function on the current group keys. This reduces the communication and computation cost since it avoids reconstruction of the polynomial.

4. Comparison

Storage at Each User. In Dexter et al. scheme [21], each user $u \in P_i$ stores the shares of the keys along the path from leaf to the root, $i = 1, 2, \ldots, M$. If there are S_i users in P_i, then height of the tree is $\lceil \log_2 S_i \rceil$. Thus each user $u \in P_i$ stores $(t - 1)\lceil \log_2 S_i \rceil$ number of elements.

In our scheme, each user $u \in P_i$ stores $(t - 1)$ prepositioned shares, an activating share, i number of group shares and $\lceil \log_2 S_i \rceil$ auxiliary keys.

Storage at KDC. KDC is required to store shares of all the keys in the system. For a total of N users in the system, there are at most $2N - 1$ nodes. Thus, storage required at KDC in Dexter et al. scheme is $(t-1)(2N-1)$. In our scheme there are $t + M - 1$ shares and $2N - M - 1$ auxiliary keys in the system. Hence KDC is required to store only $2N + t - 2$ elements.

Encryption Cost. In Dexter et al. scheme, if a member leaves any layer i, $i = 1, \ldots, M$, in order to change the keys along the path till the root, corresponding prepositioned shares must be changed, which leads to $(t - 1)\lceil \log_2 S_i \rceil$ encryptions. Also, prepositioned shares meant for different layers must be changed, which results in $(t - 1)i$ encryptions. Hence, the number of elements encrypted is $(t - 1)(\lceil \log_2 S_i \rceil + i)$. Whereas in our scheme, keys along the path and an activating share are encrypted; thus, it is just $(\lceil \log_2 S_i \rceil + it)$ encryptions.

Computation Cost. In Dexter et al. scheme [21], the group key for each layer i is computed by constructing a $(t - 1)$ degree polynomial and evaluated at 0. In our scheme, as we move up the hierarchy, degree of the polynomial is incremented by 1. Though it requires more amount of computation as compared to $(t-1)$ degree polynomial, it improves the resistance of the system to attack; hence, the system is more secure.

Table 1 gives the comparison of our scheme with the scheme proposed by Dexter et al. [21] in terms of storage and encryption cost.

Table 2 compares the performance of our scheme with Dexter et al. scheme [21]. To have fair comparison we consider 4 layers N_1, N_2, N_3, and N_4 and a polynomial of degree 5, that is, $t - 1 = 5$. N_1 is the layer at lower privilege level, and N_4 is at higher privilege level.

Consider an example with 128 users at layer N_1, 64 users each at layers N_2 and N_3, and 32 users at layer N_4. Heights of the trees at layers N_1, N_2, N_3, and N_4 are 7, 6, 6, and 5, respectively. Number of keys stored at KDC in Dexter et al. scheme is $(t - 1)(2N - 1)$ at each layer which sums up to be 2860, whereas in our scheme we get only 588 keys at the KDC which is computed as $2N + t - 2$.

In Dexter et al. scheme, users at layer N_1 store $h_1 + i = 7 + 1$ sets of prepositioned information, namely, $8 * 5 = 40$ keys. Users at layer N_2 store $6 + 2 = 8$ sets of prepositioned information, users at layer N_3 store $6 + 3 = 9$ sets of prepositioned information, and users of layer N_4 store $5 + 4 = 9$ sets of prepositioned information. In our scheme, the number of keys at different layers N_1, N_2, N_3, and N_4 is only 12, 12, 13, and 13, respectively.

Table 1: Comparison of storage and encryption cost.

	Dexter et al. scheme	Our scheme
Storage at KDC	$(t-1)(2N-1)$	$2N+t-2$
Storage at each user of layer i	$(t-1)(\lceil \log_2 S_i \rceil + i)$	$t+\lceil \log_2 S_i \rceil + i - 1$
Number of elements encrypted during leave from layer i	$(t-1)(\lceil \log_2 S_i \rceil + i)$	$\lceil \log_2 S_i \rceil + it$

Table 2: Performance of multilayer SGC.

Number of users in different layers				Number of keys at server		Number of keys at each user							
N_1	N_2	N_3	N_4	Dexter et al. scheme	Our scheme (% saving)	Dexter et al. scheme				Our scheme (% saving)			
						l_1	l_2	l_3	l_4	l_1	l_2	l_3	l_4
128	64	64	32	2860	592 (79)	40	40	45	45	12 (70)	12 (70)	13 (71)	13 (71)
4096	1024	512	64	56940	11408 (80)	65	60	60	50	17 (73)	16 (73)	16 (73)	14 (72)
2^{15}	2^{12}	64	16	369420	73904 (80)	80	70	45	40	20 (75)	18 (74)	13 (71)	12 (70)
2^{20}	2^{15}	2^{10}	2^7	10824940	2165008 (80)	105	85	65	55	25 (76)	21 (75)	17 (73)	15 (72)

In Table 2 we also have recorded the percentage of savings achieved in our scheme. From the values recorded in Table 2, it is clear that we achieve substantial savings in storage and encryption cost as compared to Dexter et al. scheme. For instance, for a secure group with 4096 users and with a fixed security parameter $t = 10$, we achieve 80% savings in storage at KDC and about 73% savings in storage at the users.

5. Multiple Simultaneous SGC

A project may be divided into several modules, and each module may be assigned to a group of members. It may be necessary for some members to deal with two or more modules depending on the requirement. It is required that each module should be developed confidentially so that members developing a particular module must communicate among themselves securely. Hence, each group should have a group key, and members belonging to two or more groups should possess group keys of all those groups for which they are members. We develop a key management scheme for such multiple SGC with efficient storage, computation, and communication costs [23].

5.1. Key Management Scheme. We consider a set of users $U = \{u_1, u_2, \ldots, u_N\}$ and M subgroups $P_1, P_2, \ldots, P_M$ such that some users are present in more than one subgroup. For each subgroup P_i, a logical key tree is constructed, $i = 1, 2, \ldots, M$. The height of the tree for subgroup P_i depends on the number of users in P_i. If there are N_i ($N_i \leq N$) number of users in group P_i, then the height is $h_i = \lceil \log_2 N_i \rceil$. An user u_i is assigned with a private key K_i, $i = 1, 2, \ldots, N$ and auxiliary keys along the path from u_i to root of the key tree. This section deals about initial group setup and computation of group key(s).

5.2. Initial Group Setup and Group Key Computation. Our scheme is based on centralized key management scheme using logical key tree (LKT) approach as proposed in [7]. Hence, we assume a trusted KDC which is responsible for initial group(s) setup and rekeying operations. Users in the system are provided with unique identification number, and the groups are assigned with group numbers. To begin with we allow the KDC to fix the security parameter t.

(1) User u_i, $i = 1, 2, \ldots, N$ who would like to join the group P_j, $j = 1, 2, \ldots, M$ sends a join request to KDC. The KDC generates and sends a unique private key K_i, $i = 1, 2, \ldots, N$ to the requesting user u_i over a secure channel (we assume that, at the initial stage, a secure channel is established between KDC and the joining user). Hence, every user shares a private key with the KDC.

(2) KDC selects randomly $t-2$ number of points (x_i, y_i) in GF(p) called *prepositioned basic shares*, $i = 1, 2, \ldots, t-2$ and (x_t, y_t) as activating share (AS). These shares are sent *securely* to the members of all the subgroups $P_1, P_2, \ldots, P_M$.

(3) KDC selects randomly M points (xg_i, yg_i) in GF(p) called *prepositioned group shares* distinct from the previously selected points and sends (xg_i, yg_i) securely to the members of the subgroup P_i, $i = 1, 2, \ldots, M$.

(4) KDC constructs LKT for the group P_i with the group key G_i. The key G_i is obtained by constructing a polynomial $P_i(x)$ of degree $t-1$ using the shares (x_i, y_i) of step 2 and the prepositioned group share (xg_i, yg_i). The group key is $G_i = P_i(0)$, $i = 1, 2, \ldots, M$. KDC sends *secretly* to each user u of P_i, all the auxiliary keys along the path of LKT from the leaf u to the root.

(5) If an user u is a member of j number of groups ($1 \leq j \leq M$), it is provided with $(t-2)$ prepositioned basic shares along with AS and j number of prepositioned group shares. It constructs j distinct polynomials. A polynomial $P_k(x)$ is constructed by using $t-1$ shares of step 2 and prepositioned group share (xg_k, yg_k), $k = 1, 2, \ldots, j$. Thus, it can construct j distinct polynomials just by using one distinct group share.

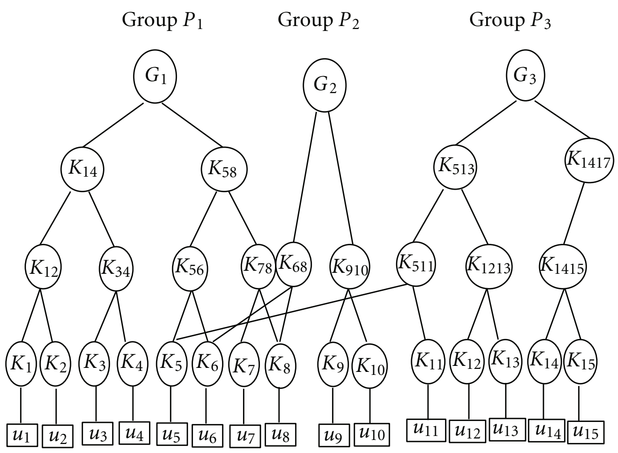

FIGURE 4: Key tree structure for multiple groups.

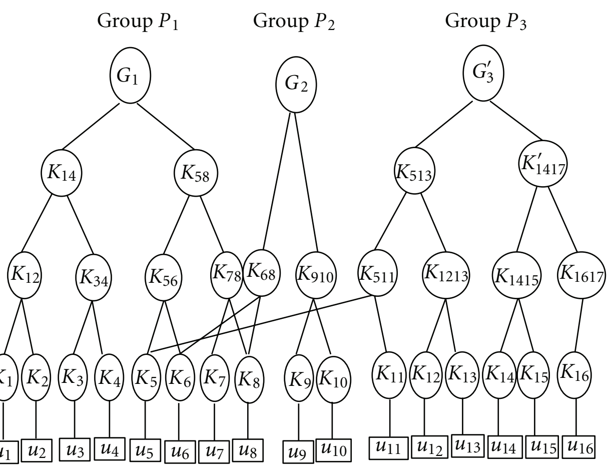

FIGURE 5: Key tree structure for multiple groups after user u_{16} joins the group P_3.

Figure 4 shows an example key tree structure with 3 groups, namely, P_1, P_2, and P_3 set up simultaneously. Group P_1 comprises of eight members $u_1, u_2, \ldots, u_8$, group P_2 contains u_6, u_8, u_9, and u_{10} as its members, whereas members u_5, u_{11}, u_{12}, u_{13}, u_{14}, and u_{15} belong to group P_3. In Figure 4, u-nodes represent users, and K-nodes represent keys. Key nodes K_1 through K_{15} are private keys of users u_1 through u_{15}, respectively, and remaining K-nodes in the figure represent auxiliary keys. G_1, G_2, and G_3 are the group keys of the groups P_1, P_2, and P_3, respectively.

For example, let us consider $t = 4$, $p = 41$ and the $(t-2)$ prepositioned basic shares as $(1, 28)$, $(2, 23)$ and AS as $(4, 4)$. Assume that KDC sends $(3, 11)$, $(3, 8)$, and $(3, 5)$ as the prepositioned group share for the members of the groups P_1, P_2, and P_3, respectively. Hence, the members of the group P_i, $i = 1, 2, 3$, now possess t; that is, 4 shares with them, and they can construct $(t-1)$ degree polynomial and evaluate it at 0 to get the group key. Thus the members of group P_1 get the group key G_1 as 14, members of P_2 get the group key G_2 as 2, and G_3 is computed by members of P_3 as 34. Hence, user U_5, for instance, can compute group keys for both the groups P_1 and P_3.

5.3. Member Join Event. If a new user u_{new} wants to join the group $P_i (1 \leq i \leq M)$, it sends a join request to KDC. KDC finds a location for the user u_{new} in the LKT of P_i and inserts it. To provide backward access control, keys along the path from the point of insertion till one level below the root are changed and communicated to the corresponding users. In order to change the group key G_i of P_i, KDC picks a new value of group share (xg_i, yg_i), encrypts it with the previous group key G_i, and sends it to the users of the group P_i. For the user u_{new}, KDC sends keys along the path from u_{new} to root, prepositioned shares (basic shares and group share) and AS after encrypting with the private key of u_{new}. The new user constructs the polynomial and evaluates it at 0 to get the group key G_i.

For instance, if a new user u_{16} sends a join request to join the group P_3, KDC inserts u_{16} at the location as shown in Figure 5. KDC changes the key K_{1417} to K'_{1417} and picks a new value for group share, say (xg'_3, yg'_3). To convey changed keys and share to corresponding users, KDC constructs the following rekey messages:

$$\begin{aligned}
&\text{KDC} \longrightarrow \{u_{14}, u_{15}\} : \{K'_{1417}\}_{K_{1415}}, \{(xg'_3, yg'_3)\}_{G_3}\\
&\text{KDC} \longrightarrow \{u_5, u_{11}, u_{12}, u_{13}\} : \{(xg'_3, yg'_3)\}_{G_3}\\
&\text{KDC} \longrightarrow \{u_{16}\} : \{K_{1617}, K'_{1417}, (xg'_3, yg'_3)\}_{K_{16}}.
\end{aligned}$$

Now, suppose that if user u_{17} sends join requests to join two groups P_2 and P_3, the LKT looks as in Figure 6 after inserting u_{17} to both the groups of Figure 5. KDC constructs the following rekey messages to convey changed keys and group shares:

$$\begin{aligned}
&\text{KDC} \longrightarrow \{u_{14}, u_{15}\} : \{K''_{1417}\}_{K_{1415}}, \{(xg''_3, yg''_3)\}_{G'_3}\\
&\text{KDC} \longrightarrow \{u_5, u_{11}, u_{12}, u_{13}\} : \{(xg''_3, yg''_3)\}_{G'_3}\\
&\text{KDC} \longrightarrow \{u_{16}\} : \{K''_{1417}\}_{K'_{1417}}, \{K'_{1417}\}_{K_{16}}, \{(xg''_3, yg''_3)\}_{G'_3}\\
&\text{KDC} \longrightarrow \{u_6, u_8, u_9, u_{10}\} : \{K_{610}, (xg'_2, yg'_2)\}_{G_2}\\
&\text{KDC} \longrightarrow \{u_{17}\} : \{K_{17-2}, K_{17-1}, K'_{1617}, K''_{1417}, (xg''_3, yg''_3)\}_{K_{17}}.
\end{aligned}$$

If a member joins a group G_i with N_i members, then at most $\lceil \log_2 N_i \rceil$ keys are changed. To convey changed keys to the members of the group, $\lceil 2\log_2 N_i \rceil$ encryptions are performed and $\lceil \log_2 N_i \rceil$ rekey messages are constructed. In general, if a member joins j number of groups, $\sum_{i=1}^{j} \log_2 N_i$ keys are changed, $2\sum_{i=1}^{j} \log_2 N_i$ encryptions are performed, and $\sum_{i=1}^{j} \log_2 N_i$ rekey messages are constructed.

5.4. Member Leave Event. A member may leave the group either voluntarily or KDC may forcibly expel the member from the group. In any case, the keys known to leaving member in the LKT must be changed to provide forward confidentiality. If a member u_l wants to leave the group $P_i (1 \leq i \leq M)$, it sends a leave request to KDC. Here, we encounter two cases.

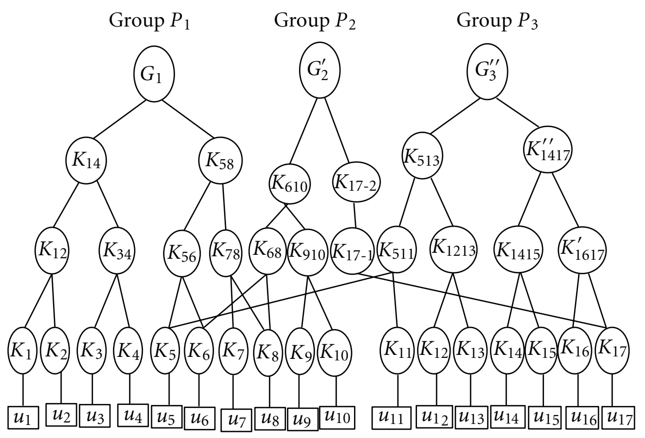
Group P_1
Group P_2
Group P_3
G_1
G_2'
G_3''
K_{14}
K_{58}
K_{610}
$K_{17\text{-}2}$
K_{513}
K_{1417}''
K_{12}
K_{34}
K_{56}
K_{78}
K_{68}
K_{910}
$K_{17\text{-}1}$
K_{511}
K_{1213}
K_{1415}
K_{1617}'
K_1 K_2 K_3 K_4 K_5 K_6 K_7 K_8 K_9 K_{10} K_{11} K_{12} K_{13} K_{14} K_{15} K_{16} K_{17}
u_1 u_2 u_3 u_4 u_5 u_6 u_7 u_8 u_9 u_{10} u_{11} u_{12} u_{13} u_{14} u_{15} u_{16} u_{17}

FIGURE 6: Key tree structure for multiple groups after user u_{17} joins the groups P_2 and P_3.

Case 1. If u_l belongs to only one group P_i,

(i) KDC removes the corresponding user-node and private key-node from LKT,

(ii) KDC changes the keys along the path from leaving point till one level below the root in P_i selects new group share (xg_i, yg_i) and conveys to corresponding users in P_i.

Case 2. If u_l belongs to more than one group,

(i) KDC detaches u_l from the group P_i,

(ii) KDC changes the keys along the path from leaving point till one level below the root in P_i selects new group share (xg_i, yg_i) and conveys to corresponding users in P_i.

For example, consider the multiple groups scenario as in Figure 6. Now, if user u_5 wants to leave the group P_3, it sends a leave request to KDC. KDC detaches u_5 from the LKT of P_3 and changes the keys along the path as shown in Figure 7, and to convey changed keys it constructs the following rekey messages:

$$\text{KDC} \rightarrow \{u_{14},\ldots,u_{17}\} : \{(xg_3''', yg_3''')\}_{K''_{K_{1417}}}$$

$$\text{KDC} \rightarrow \{u_{12},u_{13}\} : \{(xg_3''', yg_3''')\}_{K_{1113}}, \{K_{1113}\}_{K_{1213}}$$

$$\text{KDC} \rightarrow u_{11} : \{(xg_3''', yg_3''')\}_{K_{1113}}, \{K_{1113}\}_{K_{11-1}}, \{K_{11-1}\}_{K_{11}}.$$

If a member leaves the group P_i, which contains N_i members, then $\lceil \log_2 N_i \rceil$ values are changed, $\lceil 2\log_2 N_i \rceil$ number of encryptions are performed, and $\lceil \log_2 N_i \rceil$ rekey messages are constructed to convey changed keys to the members of the group.

5.5. Member Moving from One Secure Group to Another. There are two cases.

Case 1. A member u_m wants to move from group P_i to the group P_j.

(i) u_m sends a move request to KDC.

(ii) This request is interpreted as member leave event for the group P_i and member join event for group P_j.

(iii) KDC detaches u_l from the LKT of the group P_i.

(iv) To provide forward access control for group P_i, KDC changes the keys along the path from the leaving point till one level below the root in the LKT of group P_i.

(v) KDC inserts u_{new} in LKT of the group P_j.

(vi) To provide backward access control in group P_j, KDC changes the keys along the path from insertion point till the root in LKT of group P_j.

(vii) To change group keys of the groups P_i and P_j, KDC changes group shares (xg_i, yg_i) and (xg_j, yg_j).

(viii) KDC conveys securely changed keys and group shares to corresponding members of the groups P_i and P_j.

Case 2. A member $u_m \in P_i$ wants to join the group P_j.

(i) u_m sends a join request to KDC.

(ii) KDC inserts u_m in the LKT of group P_j.

(iii) To provide backward access control in group P_j, KDC changes the keys along the path from insertion point till the root in the LKT of group P_j.

(iv) To change group key of P_j, KDC changes group share (xg_j, yg_j).

(v) KDC conveys securely changed keys and group share to corresponding members of the group P_j.

To illustrate the member-moving scenario, consider Figure 7 and assume that user u_4 wants to move from group P_1 to group P_2. It sends to KDC the move request. KDC inserts u_4 in the group P_2 as shown in Figure 8 and changes the keys K_{34} and K_{14} in group P_1 and the keys K_{17-1}, K_{17-2} in group P_2. It picks new values for group shares (xg_1, yg_1) and (xg_2, yg_2), and, in order to convey changed keys and shares securely, it constructs the following rekey messages:

$$\text{KDC} \rightarrow \{u_1,u_2\} : \{K_{13}\}_{K_{12}}, \{(xg_1', yg_1')\}_{K_{13}}$$

$$\text{KDC} \rightarrow u_3 : \{K_{13}\}_{K_{3-1}}, \{K_{3-1}\}_{K_3}, \{(xg_1', yg_1')\}_{K_{13}}$$

$$\text{KDC} \rightarrow \{u_5,u_7\} : \{(xg_1', yg_1')\}_{K_{58}}$$

$$\text{KDC} \rightarrow \{u_6,u_8\} : \{(xg_1', yg_1')\}_{K_{58}}, \{(xg_2', yg_2')\}_{G_2'}$$

$$\text{KDC} \rightarrow \{u_9,u_{10}\} : \{(xg_2', yg_2')\}_{G_2'}$$

$$\text{KDC} \rightarrow u_{17} : \{K_{417-2}\}_{K_{417-1}}, \{K_{417-1}\}_{K_{17}}, \{(xg_2', yg_2')\}_{G_2'}$$

$$\text{KDC} \rightarrow u_4 : \{K_{417-2}\}_{K_{417-1}}, \{K_{417-1}, (xg_2', yg_2')\}_{K_4}.$$

Thus, when a member moves from one group with N_i members to another group with N_j members, $\lceil \log_2 N_i + \log_2 N_j \rceil$ keys are changed, $\lceil 2(\log_2 N_i + \log_2 N_j) \rceil$ encryptions are performed, and $\lceil \log_2 N_i + \log_2 N_j \rceil$ rekey messages are constructed.

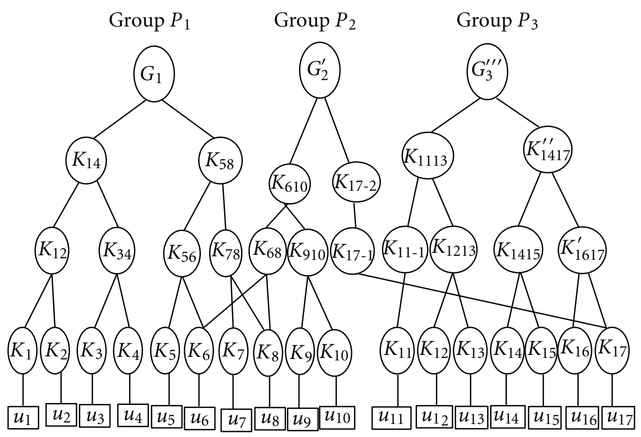
Group P_1
Group P_2
Group P_3
G_1
G_2'
G_3'''
K_{14}
K_{58}
K_{610}
$K_{17\text{-}2}$
K_{1113}
K_{1417}''
K_{12}
K_{34}
K_{56}
K_{78}
K_{68}
K_{910}
$K_{17\text{-}1}$
$K_{11\text{-}1}$
K_{1213}
K_{1415}
K_{1617}'
K_1 K_2 K_3 K_4 K_5 K_6 K_7 K_8 K_9 K_{10} K_{11} K_{12} K_{13} K_{14} K_{15} K_{16} K_{17}
u_1 u_2 u_3 u_4 u_5 u_6 u_7 u_8 u_9 u_{10} u_{11} u_{12} u_{13} u_{14} u_{15} u_{16} u_{17}

Figure 7: Key tree structure for multiple groups after user u_5 leaves the group P_3.

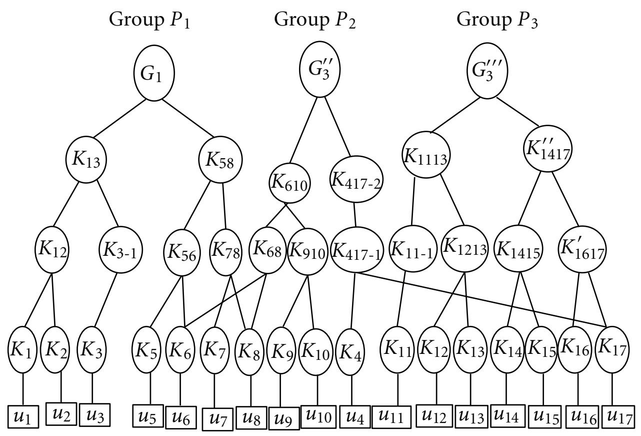
Group P_1
Group P_2
Group P_3
G_1
G_3''
G_3'''
K_{13}
K_{58}
K_{610}
$K_{417\text{-}2}$
K_{1113}
K_{1417}''
K_{12}
$K_{3\text{-}1}$
K_{56}
K_{78}
K_{68}
K_{910}
$K_{417\text{-}1}$
$K_{11\text{-}1}$
K_{1213}
K_{1415}
K_{1617}'
K_1 K_2 K_3 K_5 K_6 K_7 K_8 K_9 K_{10} K_4 K_{11} K_{12} K_{13} K_{14} K_{15} K_{16} K_{17}
u_1 u_2 u_3 u_5 u_6 u_7 u_8 u_9 u_{10} u_4 u_{11} u_{12} u_{13} u_{14} u_{15} u_{16} u_{17}

Figure 8: Key tree structure for multiple groups after user u_4 moves from group P_1 to group P_2.

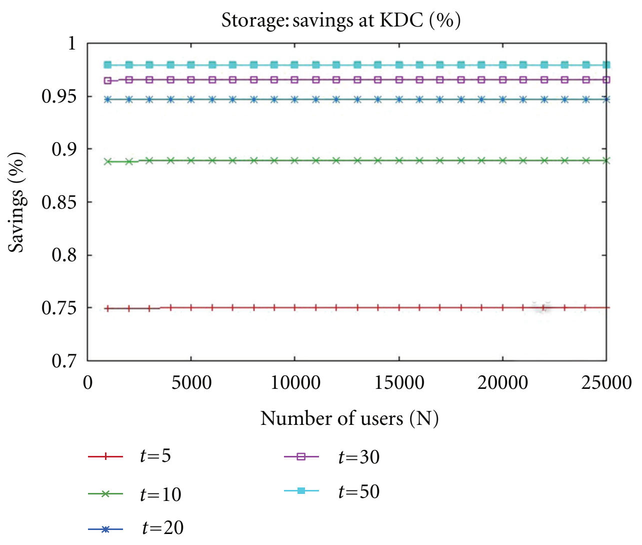

Figure 9: Storage cost: percentage savings at the KDC.

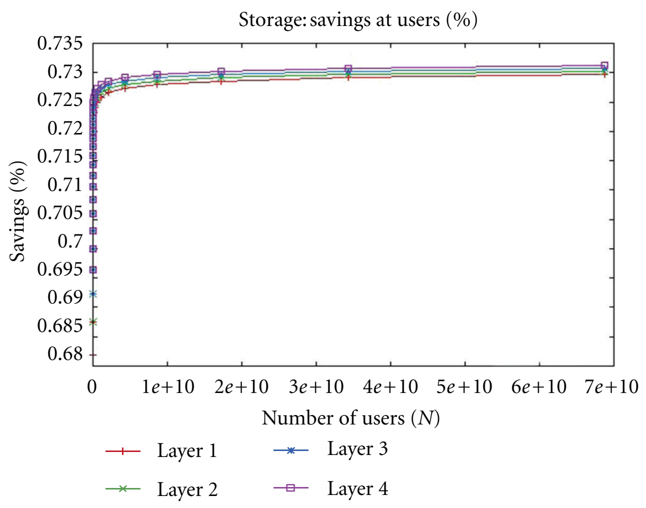

Figure 10: Storage cost: percentage savings at the users with $t = 5$.

5.6. Storage. If there are N_i users in group P_i, then the height of the LKT for P_i is $\lceil \log_2 N_i \rceil$. A user u of the group P_i stores $\lceil \log_2 \mathrm{N}_i \rceil - 1$ auxiliary keys, $(t-1)$ prepositioned shares and an AS. If a user u is a member of j number of groups, it needs to store $\sum_{i=1}^{j}(\lceil \log_2 \mathrm{N}_i \rceil - 1)$ auxiliary keys, $(t-2+j)$ prepositioned shares, and an AS. Thus, even though a particular user belongs to all the M groups in the system, it needs to store at most $\sum_{i=1}^{M}\lceil \log_2 \mathrm{N}_i \rceil + t - 2$ elements from GF(p) and can compute keys for all the groups.

We plot the graphs to depict the percentage of savings achieved in storage cost and encryption cost when compared to Dexter et al. scheme. The graph in Figure 9 shows the percentage of savings achieved in storage at KDC. It is plotted for different values of the security parameter t. Figures 10 and 11 show the percentage of savings with users in different layers. They are plotted by keeping the value of t as 5 and 10, respectively. From the graphs it is clear that the storage savings at KDC varies from 75% to 95% and with the users it varies from 68% to 73%. Figures 12 and 13 show the percentage savings in encryption cost that are plotted by keeping the value of t as 5 and 10, respectively. Savings in encryption cost vary from 45% to 85%, and it is observed that the the percentage of savings is proportional to the value of t. As we move from lower layers to higher layers, the cost of savings decreases.

6. Authenticated Secure Group Communication

Once the groups are set up, members of the group can communicate with each other securely. When a member u_i, $i = 1,\ldots,N$ of group P_j, $j = 1,\ldots,M$ sends an encrypted message to members of P_j, they must identify that the message is from u_i and also if any other user u_q tries to act as u_i, others must identify that it is not u_i. This section briefs about the authenticated secure group

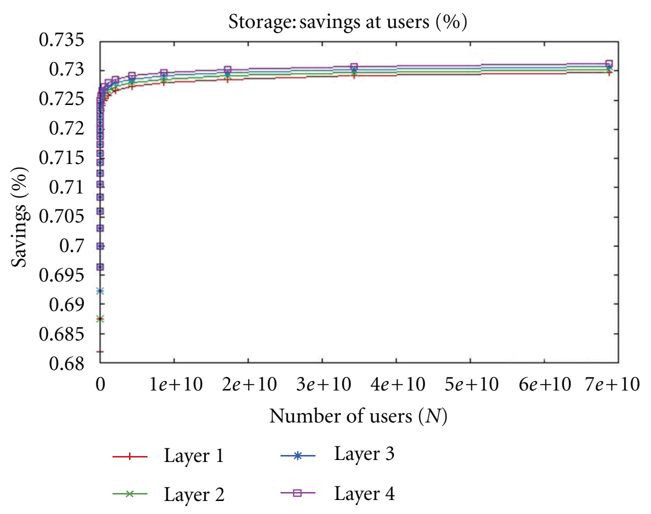

FIGURE 11: Storage cost: percentage savings at the users with $t = 10$.

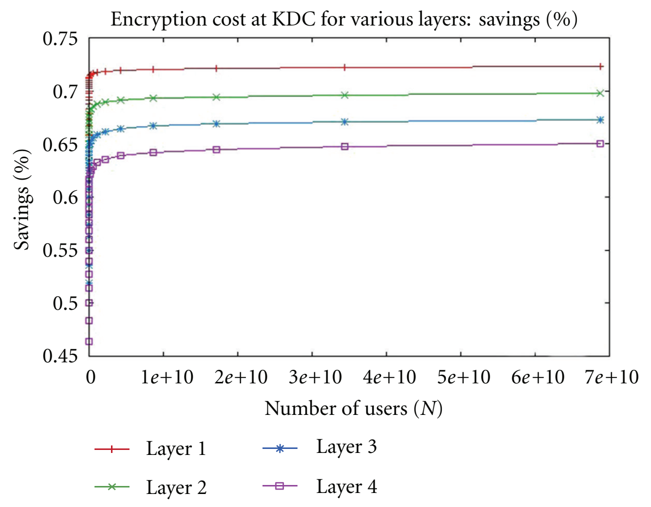

FIGURE 12: Encryption cost: percentage savings with $t = 5$.

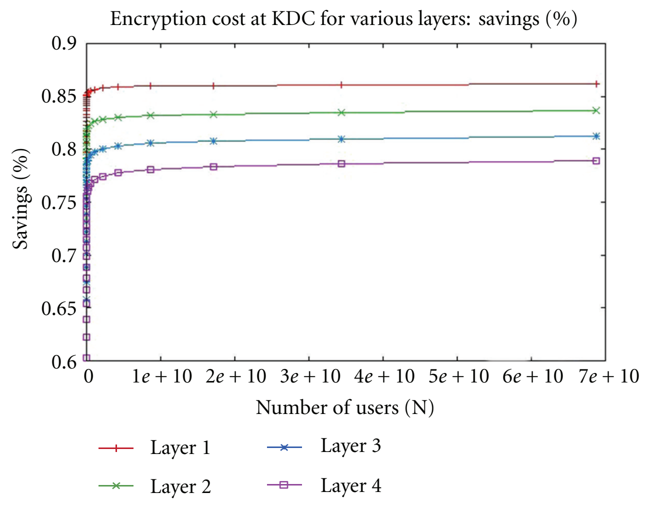

FIGURE 13: Encryption cost: percentage savings with $t = 10$.

TABLE 3: Protocol for authenticated secure group communication.

(1)	$u_i \rightarrow$ KDC: E_{K_i} [ID$_i \| P_j \| T$]
(2)	KDC $\rightarrow u_i$: E_{K_i} [$r \| P_j \| T$]
(3)	KDC $\rightarrow$ {Members of P_j}: E_{G_j}[$H(r) \|$ID$_i \| P_j \| T$]
(4)	$u_i \rightarrow$ {Members of P_j}: E_{G_j} [$r \|$ID$_i \| P_j \| T \| m$]

communication. Protocol in Table 3 depicts authenticated communication between group members. In the protocol, the symbol $\|$ denotes concatenation, and $E_K[m]$ denotes message m encrypted with key K.

If user u_i, $i = 1, \ldots, N$ wants to send a message to group members, it sends a request to KDC. Request includes identity ID$_i$ of u_i, group number P_j, $j = 1, \ldots, M$ and a time stamp value T. KDC picks a random number r from GF(p), applies hash function to compute $H(r)$, and broadcasts $[H(r) \| \mathrm{ID}_i \| P_j \| T]$ after encrypting with group key G_j, so that only the members of group P_j can decrypt it. KDC sends u_i, the message, $[r \| P_j \| T]$ after encrypting it with the private key of u_i. Thus, the value of r is available only to u_i. Now, u_i in order to send a message, m, constructs the message $[r \| \mathrm{ID}_i \| P_j \| T \| m]$, encrypts it with the group key G_j, and sends. Only members of group P_j can decrypt it and apply hash function for the received value of r to compute $H(r)$ and verify that this value is same as the one received from KDC. If it is true, then they realize that the message is from u_i as it is claimed; otherwise, they realize that some one else is trying to impersonate as u_i.

7. Conclusion

Managing multiple groups with overlapped membership is one of the important issue in group communication scenario. In this paper we proposed a scheme for such hierarchical group key management using a combination of key-based and share-based approach. It is possible for the members at higher layers to compute the keys for its own layer along with all its descendant layers just by storing extra prepositioned information. Our scheme is secure, even if a member compromises, it is not possible to get the group key unless activating share is obtained. We reduce both storage and encryption cost compared to Dexter et al. scheme. We proposed two schemes for periodic rekeying.

Managing group keys for independent simultaneous secure groups is an important issue in SGC. In this paper we considered such multiple secure groups with overlapped membership and proposed a key management scheme using a combination of key-based and share-based approach. We showed that, even if a particular user belongs to all M secure groups, it needs to store at most $(Mh + t - 2 + M)$ elements from GF(p) and is able to compute keys for all the groups. Encryption cost and number of key changes are of the order of $\log N$ for membership changes (join, leave, and a member moving from one group to another). We also provided authentication for the messages communicated between group members.

References

[1] L. R. Dondeti, S. Mukherjee, and A. Samal, "Survey and comparison of secure group communciation protocols," Tech. Rep., University of Nebraska-Lincoln, 1999.

[2] T. Hardjono and G. Tsudik, "IP multicast security: issues and directions," *Annales De Taleum*, vol. 55, no. 7, pp. 324–340, 2000.

[3] M. J. Moyer, J. R. Rao, and P. Rohatgi, "A Survey of security issues in multicast communications," *IEEE Network*, vol. 13, no. 6, pp. 12–23, 1999.

[4] Y. Amir, Y. Kim, C. Nita-Rotaru, J. L. Schultz, J. Stanton, and G. Tsudik, "Secure group communication using robust contributory key agreement," *IEEE Transactions on Parallel and Distributed Systems*, vol. 15, no. 5, pp. 468–480, 2004.

[5] Y. Amir, C. Danilov, M. Miskin-Amir, J. Schultz, and J. Stanton, "The spread toolkit: architecture and performance," Tech. Rep. CNDS-2004-1, Johns Hopkins University, 2004.

[6] M. Waldvogel, G. Caronni, D. Sun, N. Weiler, and B. Plattner, "The versakey framework: versatile group key management," *IEEE Journal on Selected Areas in Communications*, vol. 17, no. 9, pp. 1614–1631, 1999.

[7] C. K. Wong, M. Gouda, and S. S. Lam, "Secure group communications using key graphs," *IEEE/ACM Transactions on Networking*, vol. 8, no. 1, pp. 16–30, 2000.

[8] C. K. Wong, S. S. Lam, and Keystone, "A group key management service," in *Proceedings of International Conference on Telecommunications*, Acapulco, Mexico, May 2000.

[9] D. A. McGrew and A. T. Sherman, "Key establishment in large dynamic groups using one-way function trees," *IEEE Transactions on Software Engineering*, vol. 29, no. 5, pp. 444–458, 2003.

[10] S. Rafaeli and D. Hutchison, "A survey of key management for secure group communication," *ACM Computing Surveys*, vol. 35, no. 3, pp. 309–329, 2003.

[11] C. Blundo, A. de Santis, A. Herzberg, S. Kutten, U. Vaccaro, and M. Yung, "Perfectly secure key distribution for dynamic conferences," in *Advances in Cryptology-CRYPTO'92*, vol. 740 of *Lecture Notes in Computer Science*, pp. 471–486, 1993.

[12] S. M. Iolus, "A framework for scalable secure multicasting," in *Proceedings of the ACM SIGCOMM*, vol. 27, pp. 277–288, ACM, NewYork, NY, USA, September 1997.

[13] A. Fiat and M. Naor, "Broadcast encryption," in *Proceedings of 13th Annual International Cryptology Conference (CRYPTO '93)*, D. R. Stinson, Ed., pp. 480–491, August 1993.

[14] 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.

[15] D. Wallner, E. Harder, and R. Agee, "Key Management for Multicast: Issues and Architectures," Request For Comments (Informational) 2627, Internet Engineering Task Force, June 1999.

[16] A. M. Eskicioglu and M. R. Eskicioglu, "Multicast security using key graphs and secret sharing," in *Proceedings of the Joint International Conference on Wireless LANS and Home Networks and Networking*, pp. 228–241, Atlanta, Ga, USA, August 2002.

[17] A. M. Eskicioglu, S. Dexter, and E. J. Delp, "Protection of multicast scalable video by secret sharing: simulation results," in *Security and Watermarking of Multimedia Content V*, Proceedings of SPIE, Santa Clara, Calif, USA, January 2003.

[18] H. R. Hassan, A. Bouabdallah, H. Bettahar, and Y. Challal, "An efficient key management algorithm for hierarchical group communication," in *Proceedings of the 1st International Conference on Security and Privacy for Emerging Areas in Communications Networks (SECURECOMM '05)*, pp. 270–276, grc, September 2005.

[19] Y. Sun and K. J. R. Liu, "Scalable hierarchical access control in secure group communications," in *Proceedings of 23rd Annual Joint Conference of the IEEE Computer and Communication Societies (INFOCOM '04)*, March 2004.

[20] Y. Sun and K. J. R. Liu, "Multi-layer key management for secure multimedia multicast communications," in *Proceedings of International Conference on Multimedia and Expo (ICME '03)*, July 2003.

[21] S. Dexter, R. Belostotskiy, and A. M. Eskicioglu, "Multi-layer multicast key management with threshold cryptography," in *Proceedings of the Security, Steganography and Watermarking of Multimedia Content VI*, San Jose, Calif, USA, January 2004.

[22] R. Aparna and B. B. Amberker, "Key management scheme for multi-layer secure group communication," in *Proceedings of the 1st International Conference on Communication Systems and Networks (COMSNETS '09)*, January 2009.

[23] R. Aparna and B. B. Amberker, "Key management scheme for multiple simultaneous secure group communication," in *Proceedings of the IEEE International Conference on Internet Multimedia Services Architecture and Applications (IMSAA '09)*, December 2009.