Military Technical College
Kobry El-kobbah,
Cairo, Egypt

5th International Conference
on Electrical Engineering
ICEENG 2006

Hybrid Key Management for Group Communications‡

Ahmed Abdel-Hafez*,Ali Miri , Luis Orozco - Barbosa**, and Ahmed Abdel- Rahman*****

Abstract

Due to the increased popularity of group oriented applications and protocols, securing group communications has become a critical networking issue and has received much attention in recent years. A secure and efficient group key management protocol is the most fundamental challenge in group communication security. While key transport protocols may be appropriate for key establishment in large networks, many collaborative applications require distributed key agreement protocols. Proposals for key agreement protocols that have been published so far does not scale for large size group. In this paper we propose a novel framework for scalable key management protocols in group communication, using both Key Agreement and Key transport protocols. Our framework is based on a particular clustering of the members of the secure communicating group into subgroups. We describe a protocol to achieve this clustering scheme. We describe the architecture and operation of this framework using GDH.2 as a building block. We show that our framework is scalable to large groups with frequent membership changes.

1 Introduction

With the widespread use of the Internet, The popularity of *Group communication* based applications has grown considerably. *Group communication* is a means of providing multi-point to multi-point communication by organizing processes in groups. Current group-oriented applications include Internet video transmission, stock quotes, news feeds, software updates, live multi-party conferencing, online video games, collaborative workspaces and traversal of insecure networks, basic security services – such as traffic integrity, entity authentication, and

*Egyptian Armed Forces
**School of Information Technology and Engineering University of Ottawa, Ontario, Canada

***Higher Technological Institute, 10th of Ramadan
‡Extended version of the first result published in CWIT03, Ontario, Canada.

confidentiality –are necessary for these collaborative applications. These security services can be facilitated if the authorized group members share a common secret (which known in literature as a group or session
key). Thus, the fundamental service and a design challenge in secure and reliable group communication systems is the group key management. A key management protocol is a process whereby a shared secret becomes available to two or more authorized parties, for subsequent cryptographic use. Key management can be broadly subdivided into:

1. Key transport protocol (centralized key management): it is a key establishment technique in which one party creates or otherwise obtains a secret value, and securely transfers it to the other(s).

2. Key agreement protocol (contributory key management): is a key establishment technique in which two (or more) parties contribute information, which jointly establishes the shared secret key, (ideally) such that no party can predetermine the resulting value.

The collaborative applications have such diverse characteristics that it is difficult to find a unified security solution that accommodates all applications. There are many parameters that characterize these applications, which help in determining which security architecture should be used. These parameters are summarized as following : (for more discussion see [12])

- Group size
- Membership Characteristic
- Membership Dynamic
- Membership Control
- Life time
- Number and type of senders
- Volume and type of traffic

There are many other parameters which specify how the underlying communication group protocol provides membership services and reliable multicast services. The task of a membership

services is to maintain a list of the currently active and connected processes in the group. The output of the membership service is called a *view*. The reliable multicast service delivers messages to the current view members (for more details about Group Communication Specifications see [7]). Most of the proposals for group communication security that have been published so far have concentrated on the centralized key management. While the centralized approach works reasonably well for static groups or very large groups, it turns out that contributory key agreement is superior for Collaborative applications with dynamically changing membership. The reasons are as follows:

- Centralization violates the peer nature of the group by concentrating all the key generation and authentication in a single point
- A centralized key server becomes both a single point of failure or performance bottleneck and an attractive attack target.
- Environments with no hierarchy of trust are a poor match for centralized key transport.
- There is no way to guarantee the freshness and randomness of the generated key.

From the security point of view, the aforementioned reasons affect the security of the data manipulated between group members, so we focus on contributory key agreement protocol. On the other hand, there is common deficiency in the existing key agreement protocols which is it dose not scale well for large size groups (see for example [6], [11], [10], [4]). To solve this contradiction, or in other words, to efficiently extend the existing key agreement protocols to scale for large size groups (or to reduce the required cost for the small group size), we make a merge between key agreement and key transport protocols to establish a secure group session key. Where the cost for generation and regeneration of this session key will not increase linearly with the group size n.

Although we focus on contributory key agreement, we still need a central server (Network Administrator NA) to control the group membership services such as adding or deleting members and controlling the indexing scheme of the group members. While the existence of this server is vital in our protocol, it is still a matter of policy.

The rest of this paper is organized as follows. Section 2 introduces our notations and terminologies. Section 3 presents a summary of some related work and describes in a brief the

GDH.2 as it will be our building block. In section 4 we explain the main idea of our protocol and explain the different protocols which satisfy the architecture of our framework. A complexity analysis and comparison with GDH.2 and TGDH are presented in section 5. Finally we conclude with a summary of our approach and some points for future work in section 6.

2. Notations

The following notation will be used throughout the paper.

n	Number of participants in the protocol.
d	Maximum number of subgroup members.
l	The total number of levels.
i, v, j	indices of levels, subgroups and members.
i	$1, 2, \ldots l$.
v	$1, 2, \ldots d(d-1)^{i-2}$.
j	$(v-1)(d-1)+1, \ldots, v(d-1)$.
$S_{i,v}$	The v^{th} subgroup in the i^{th} level.
$M_{i,j}$	The j^{th} member in the i^{th} level.
$K_{i,v}$	The session key of the subgroup $S_{i,v}$.
N_S	Maximum number of subgroups.
E_{xT}	Total number of exponentiations.
E_{xs}	No. of exponentiations per subgroup.
BW_M	Maximum Bandwidth

The total number of group members (*n)* as a function of maximum number of subgroup members *d* and the total number of levels *l* can be formulated as:

$$n = d\sum_{i=0}^{l-1}(d-1)^i = d\frac{(d-1)^l - 1}{d-2} \qquad (1)$$

Similarly, the maximum number of subgroups, N_S in the hierarchy can be calculated as a function of (*d, l*) as follows:

$$N_S = 1 + d\frac{(d-1)^{l-1} - 1}{d-2} \qquad (2)$$

The relation between *n* and N_S can be formulated as:

$$N_S = n - d(d-1)^{l-1} + 1 \quad (3)$$

The total number of exponentiations per subgroup is based on the key agreement protocol used within the subgroup. In the case of GDH.2,

$$E_{x_s} = \frac{d^2 + 3d}{2} - 1. \quad (4)$$

Associated with these notations, we can define the following two functions:

- $Keyset(M_{i,j}) = \{K_{x,y},$ Where $1 \le x \le i$ and $y = \left\lceil \frac{j}{(d-1)^{i-x+1}} \right\rceil \}$

  $Keyset(M_{i,j})$ is the set of keys that are held by the member $M_{i,j}$. For example (refer to Figure 1) $Keyset(M_{3,8}) = \{K_{3,4}, K_{2,2}, K_{1,1}\}$. Note that each member holds its subgroup session key and its upper level (parents) subgroups session keys up till the root subgroup session key $K_{1,1}$ which is the group session key.

- $Userset(K_{i,j}) = \{M_{x,y},$ Where $(i-1) \le x \le l$ and $(j-1)(d-1)^{x-i+1} + 1 \le y \le j(d-1)^{x-i+1}\}$.

  $Userset(K_{i,j})$ is the set of users who hold the key $K_{i,j}$. Note that $K_{1,1}$ as a group session key should be held by all the members of the group. Also referring to Figure 1, $Userset(K_{2,2}) = \{M_{1,2}, M_{2,3}, M_{2,4}, M_{3,5}, M_{3,6}, M_{3,7}, M_{3,8}\}$. Note that each key, $K_{i,j}$, is held by its subgroup, $S_{i,j}$, members and its lower level (*children*) subgroups members.

3. Related work

In this section, we review several prior proposals for multiparty key agreement protocols. None of them (except TGDH [19]) provides a solution to the scalability problem. More precisely, the cost of generation of a group session key is proportional to the group size *n* (*O*(*log n*) in case of TGDH). The earliest attempt to provide contributory group key agreement, which extends 2-party DH protocol [18] to a group of size *n* is due to Ingemarsson et al. [9]. The protocol requires synchronous startup and executes in (*n* ¡ 1) rounds. The members must be arranged in a logical ring. In a given round, every participant raises the previously received intermediate key value to the power of its exponent and forwards the result to the next participant. After (*n* ¡ 1) rounds everyone has computed the same key K_n.

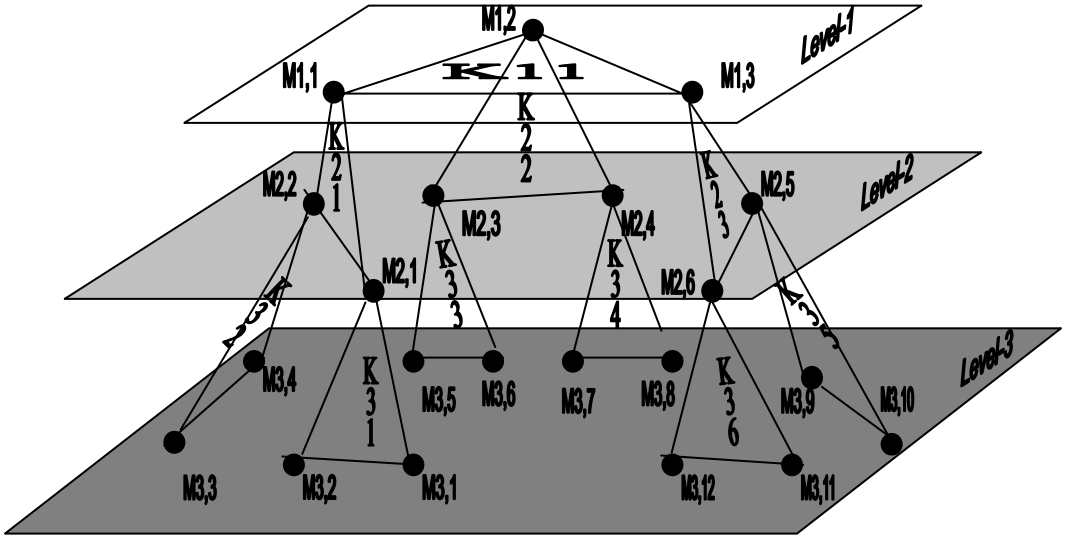

Figure 1 Members Distribution

This work did not address the case of rekeying after membership changes. Another interesting scheme (STR) was presented by Steer et al. in [3]. This protocol requires all members to have broadcasting facilities and takes *n* rounds to complete. In some ways STR is similar to GDH.2, but the key in STR has a very different structure: $K_n = \alpha^{N_n \alpha^{N_{n-1} \alpha^{\ldots^{N_3 \alpha^{N_1 N_2}}}}}$. Interestingly, STR is well suited for adding new members. Member deletion, on the other hand, is difficult in STR since there is no natural group controller.

The TGDH protocol [19] is the only protocol that addresses the scalability issue in key agreement protocol. TGDH is an adaptation of key trees in the context of fully distributed, contributory group key agreement. The tree is organized in the following manner: each node $\langle l,v \rangle$ is associated with a key $K_{\langle l,v \rangle}$ and the corresponding blinded key $BK_{\langle l,v \rangle} = g^{K}{}_{\langle l,v \rangle} \bmod p$. The key at the root node is the group key shared by all members, and a key at the leaf node is the random session contributed by a group member. The basic idea is that every member can compute a group key when all blinded keys on the key tree are known. After any group membership event, every member unambiguously adds or removes some nodes related with the event, and invalidates all the affected nodes.

Steiner *et al.* have addressed the dynamic membership issues in their group key agreement proposal [7] and have proposed a family of Group Diffie-Hellman (GDH) protocols (namely, GDH.1, GDH.2, and GDH.3). Their protocols based on straight forward extension of the two-party Diffie-Hellman. GDH provides contributory authenticated key agreement, key independence, key integrity, resistance to known key attacks, and perfect forward secrecy. Their protocol suite is fairly efficient in leave and partition operation, but the merge protocol requires as many rounds as the number of new members to complete the key agreement procedure.

3.1 Group Diffie-Hellman GDH.2

In this section we briefly describe the Group Diffie-Hellman protocol GDH.2 (see Figure 2) proposed in [7]. The protocol consists of two stages, upflow and downflow. The purpose of the upflow stage is to collect contributions from all group members. As shown in Figure 2, *Mi* receives the contribution from $M_{i,1}$ and compose *i* intermediate values (each with (*i* - 1) exponents) and one cardinal value containing *i* exponents. For example, M_4 receives a set: $\left\{\alpha^{N_1N_2N_3}, \alpha^{N_1N_2}, \alpha^{N_1N_3}, \alpha^{N_2N_3N_4}\right\}$ and outputs the set: $\left\{\alpha^{N_1N_2N_3N_4}, \alpha^{N_1N_2N_3}, \alpha^{N_1N_2N_4}\alpha^{N_1N_3N_4}, \alpha^{N_2N_3N_4}\right\}$. The cardinal value in this example is $\alpha^{N_1N_2N_3N_4}$. By the time the upflow reaches M_n, the cardinal value becomes $\alpha^{N_1\ldots N_{n-1}}$. Thus M_n is the first group member to compute the key K_n. Also, as the final part of the upflow stage, M_n computes the last batch of intermediate values. In the second stage M_n, *broadcasts* the intermediate values to all group members.

4. Our Approach

We present in this paper a new approach to improve the scalability of key agreement protocols. By the scalability problem in key agreement protocols for group communication we address: how can we (efficiently) generate/regenerate a group session key within a large group size and/or a group with frequent membership changes?

Our hierarchical scheme based on distribution of group members into particular size-bounded subgroups at different levels. Our motivation for this approach is that the key generation/regeneration cost (communication and computation overhead) of the most efficient key agreement protocols that have been published is proportional to the group size *n*. For large size

groups, the existing protocols are not efficient. Using the hierarchical system, the cost of generating/regenerating the group session key increases linearly with the logarithm of the group size *n* (or less). This improvement comes from using parallel processing for the same level's subgroups concurrently, which reduces the communication and/or the computation overheads needed to generate/regenerate the group session key.

4.1Members Distribution

The distribution of the group members in a hierarchical scheme, as shown in Figure 1, consists of many levels which permits the key agreement protocol to scale to large number of members. Each level consists of one or more subgroups. Levels are numbered sequentially starting from the top level (level-1) and ending at the (level-*l)*. The subgroup $S_{1,1}$ is the root subgroup consists of *d* members. To construct our hierarchical distribution, each member $M_{1,i}$ (where i = 1,…., d) of the $S_{1,1}$ subgroup, with another (d-1) members, will constitute level-2 subgroups $S_{2,i}$. Each member $M_{2,j}$ (where j = 1,…, $d(d$-1)) of the level 2 subgroups, with another (d-1) members will constitute the level 3 subgroups $S_{3,j}$.

This process will continue till all members of the group participate in the hierarchical system. The maximum number of levels will be *l*.

The indexing scheme of members (subgroups) determines the position of any member (subgroup) in the hierarchy based on the maximum number of members *d*, in each subgroup. For example, if we have a hierarchical system with d = 3 members in each subgroup, $M_{3,12}$ is the 12^{th} member in the 3^{rd} level (see Figure 1).

Also from this indexing scheme we can determine the subgroup leader (which is chosen by the *NA*), parent subgroups up to the root level subgroup and also the children subgroups down to the level *l*.

The indexing scheme of members (subgroups) determines the position of any member(subgroup) in the hierarchy based on the maximum number of members *d*, in each subgroup. For example, if we have a hierarchical system with d = 3 members in each subgroup, *M*3;12 is the 12*th* member in the 3^{rd} level (see figure 1). Also from this indexing scheme we can

determine the subgroup leader (which is chosen by the NA), parent subgroups up to the root level subgroup and also the children subgroups down to the level *l*.

The number of keys that are held by any member $M_{i,j}$ is equal to *i* + 1, where *i* is the level to which it belongs. For example (also for *d* = 3), the member $M_{2,6}$ holds the set of keys $keyset(M_{2,6}) = (\ K_{3,6}, K_{2,3}, K_{1,1}\ )$.

4.2 Initial Key Generation Protocol

In the following protocols, we use GDH.2 as a building block to generate/regenerate the keying material (Note that our hierarchical system can use any key agreement protocol as a building block). In this section we will explain how our protocol can extend GDH.2 to work efficiently with large size group. The protocol starts from the lowest level *l*. The members of each subgroup, $S_{l,x}$ (where $x \in [1, v]$, and $v = d(d-1)^{l-2}$), separately and collaboratively generate their session key $K_{l,x}$ using GDH.2 protocol. This process runs concurrently for all subgroups $S_{l,x}$. After agreeing on the subgroup session key $K_{l,x}$, the members of each subgroup in the upper level $S_{l-1,\bar{x}}$ (where $\bar{x} \in \left[1, \frac{v}{d-1}\right]$ will generate their session keys, $K_{l-1,\bar{x}}$. After agreeing on the subgroup session key, each member in the subgroup $S_{l-1,\bar{x}}$ (except the subgroup leader $M_{l-2,\bar{x}}$) will broadcast the generated subgroup session key $K_{l-1,\bar{x}}$ to its *l*-level subgroup's members encrypted by the *l*-level subgroup session key. This process will continue until the subgroup leaders in level 2, which constitute the root subgroup, $S_{1,1}$, generate the level-1 session key, $K_{1,1}$, which is the group session key, and then each member in the subgroup $S_{1,1}$ will broadcast the group session key $K_{1,1}$, encrypted with its level-2 session key $K_{2,y}, (y = 1,2,...,d)$ to its sibling members. The initial key generation protocol is depicted in Figure 3. To illustrate our approach, we will explain the protocol using a simple numerical example, which is clearly depicted in Figure 1. Suppose we have 21 users willing to participate in a communication group and we would like to generate a session key to enable all members to securely communicate (we can pick any number to make a group but we chose this number for simplicity to constitute a balanced hierarchical system). Suppose we choose

the number of subgroup members to be, $d = 3$ and the maximum number of levels to be $l = 3$. (We can choose a different value of d and subsequently l, but these values will keep the balance of the hierarchical system). After the proper subgroup constitution is done as explained in section 4.1 and every member has been assigned a certain index according to its level and its number in that level, the key generation process can begin.

To generate the group session key, the protocol starts from level 3. As mentioned before, each subgroup members at level 3, $S_{3,x}$ (where $x = 1,\ldots, 6$) will generate their subgroup session key $K_{3,x}$. For example, the subgroup $S_{3,4}$, which consists of members $\{M_{2,4}, M_{3,7}, M_{3,8}\}$, will generate the key $K_{3,4}$. The subgroup leader $M_{2,4}$ with the other members in the subgroup $S_{2,2} = \{M_{1,2}, M_{2,3}\}$ will generate the subgroup session key $K_{2,2}$. Each member of $S_{2,2}$ (except $M_{1,2}$) will broadcast key $K_{2,2}$ to the level 3 members encrypted by the level 3 subgroup session keys. For example:

$$M_{2,4} \Rightarrow \{M_{3,7}, M_{3,8}\} : (K_{2,2})_{K_{3,4}}.$$

$$M_{2,3} \Rightarrow \{M_{3,5}, M_{3,6}\} : (K_{2,2})_{K_{3,3}}.$$

At the same time $M_{1,2}$ with $M_{1,1}, M_{1,3}$, (which constitutes the root level subgroup $S_{1,1}$) will generate the group session key $K_{1,1}$ and broadcast it to the rest of the group members encrypted with the lower level subgroup session key. For example:

$$M_{1,1} \Rightarrow \{M_{2,1}, M_{2,2}, M_{3,1}, M_{3,2}, M_{3,3}, M_{3,4}\} : (K_{1,1})_{K2,1}.$$

$$M_{1,2} \Rightarrow \{M_{2,3}, M_{2,4}, M_{3,5}, M_{3,6}, M_{3,7}, M_{3,8}\} : (K_{1,1})_{K2,2}.$$

$$M_{1,3} \Rightarrow \{M_{2,5}, M_{2,6}, M_{3,9}, M_{3,10}, M_{3,11}, M_{3,12}\} : (K_{1,1})_{K2,3}.$$

The proposed hierarchical protocol has the following characteristics for initial key generation, where the maximum number of subgroup members is d, the maximum number of levels is l and the number of subgroups is N_S.

Rounds	$R = l(d+1) - 1$
Messages	$M = d(N_S + 1)$
Exponentiations	$Ex_T = Ex_s(N_S)$
Encryptions	$E_n = N_S - 1$
Max. Bandwidth	$BW_M = d$

4.3Group Membership Changes

In the previous section, we have assumed that the group membership is determined in advance and that all the members have been authorized to share in the group communication prior to the execution of the key generation protocol. However, due to the dynamic nature of group communication, our key agreement protocol must handle adjustments to group secrets after any membership change operation in the underlying group communication system. The membership changes include new member(s) joining in and/or old member (s) leaving (or getting evicted from) the group. Moreover, due to environmental factors such as network failure, groups can be partitioned.

Similarly, when network partitions heal, multiple groups need to merge into one. In addition to the aforementioned membership operations, periodic refreshes of group secrets are advisable to limit the amount of ciphertext generated with the same key and to recover from potential compromises of members' contributions of prior session key. In the following sections, we will explain how our protocol supports the mentioned operations (join, leave, merge, partition, key refresh). In the following sections (4.3.1 to 4.3.5), we assume that every member can determine the insertion point in the hierarchy (the subgroup where this member will join). Note that the key refresh operation is, in fact, a special case of the leave operation, without any member leaving the group, where the root subgroup $S_{1,1}$ regenerate a new group session key $K_{1,1}$ and broadcast it to the other group members encrypted with the suitable lower level key. Moreover, any subgroup can refresh its subgroup session key and proceed in the protocol as previously mentioned.

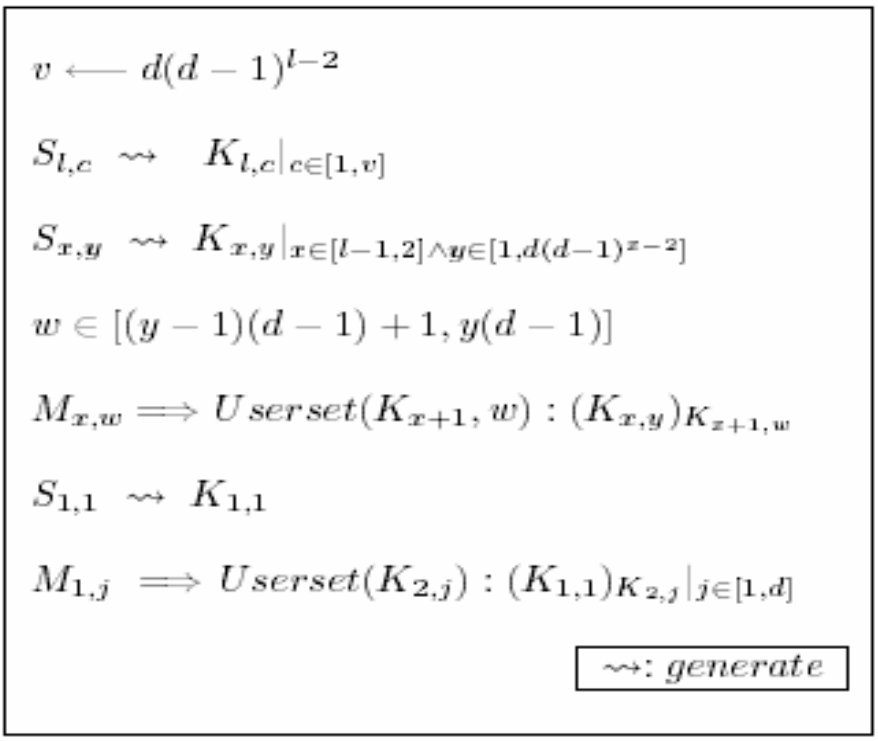

Figure 3: Initial key generation protocol

4.3.1 Join protocol

The main security requirement of member join is the secrecy of the previous group keys with respect to both outsiders and new group member(s). In our protocol this can be achieved as follows (Figure 4): A user who wants to join the group initiates the protocol by sending a join request message to the network administrator *NA* who checks the eligibility of the new member to join this group. If the check passes, *NA* replies with the needed cryptographic public parameters *CP* (according to the used protocol) and also the specific insertion point (a specific subgroup $S_{i,v}$ and its number in the subgroup). At the same time *NA* sends to the members of the subgroup, where the new member will join, a message to notify them that there is a new member will join the group with its information (the index of the new member which specifies its position in the subgroup). A new member can join at any level which keeps the hierarchical balance i.e., if there is any subgroup that contains a number of members < *d*, the new member will join this subgroup (to improve the efficiency, the preference will be to the upper levels). On the other hand, if the

distribution is well balanced, the new member joins at any subgroup at any level, the preference also be to the upper layer, the best is to join at the root subgroup $S_{1,1}$. After the new member has joined to a certain subgroup, for example, $S_{i,v}$ the new member index will be $M_{i,v(d-1)+1}$ or $M_{1,d+1}$ in case of joining $S_{1,1}$. This subgroup (including the new member) will regenerate a new subgroup session key $\bar{K}_{i,v}$. The old members $(M_{i,x})$ of the subgroup $S_{i,v}$ (where $x \in [(v-1)(d-1)+1, v(d-1)]$ broadcast the new session key to the *i*+1 subgroups $S_{i+1,x}$ encrypted with the keys $K_{i+1,x}$. At the same time, the parent level subgroup, $S_{i-1,z}, \left(z = \left\lceil \frac{v}{d-1} \right\rceil\right)$ will regenerate a new subgroup session key $\bar{K}_{i-1,z}$ and also broadcast the new session key $\bar{K}_{i-1,z}$ to the lower level subgroups. This procedure will continue up to the root level subgroup key $\bar{K}_{i,1}$ is regenerated and broadcasted to the other members. We have to note that all the down broadcasting messages to the *children* should be encrypted with the *Keyset*(*children*).

From the previous description of the join protocol, it is clear that the overhead cost of the protocol (communications and computations) needed to regenerate the session key depends on the position where the new member joins. The minimum cost of regeneration will be if the new member joins at the root level subgroup. On the other hand, if the new member joins at level *l* it will cost the maximum number of operations. The following characteristics
are the maximum values i.e., if the new member joins at the *l*-level.

Rounds	$R = 2l$
Messages	$M = 2 + d(l-1)$
Exponentiations	$Ex_T = d + 3 + l(2d-1)$
Encryption	$E_n = d + (d-1)(l-2)$
Max. Bandwidth	$BW_M = d$

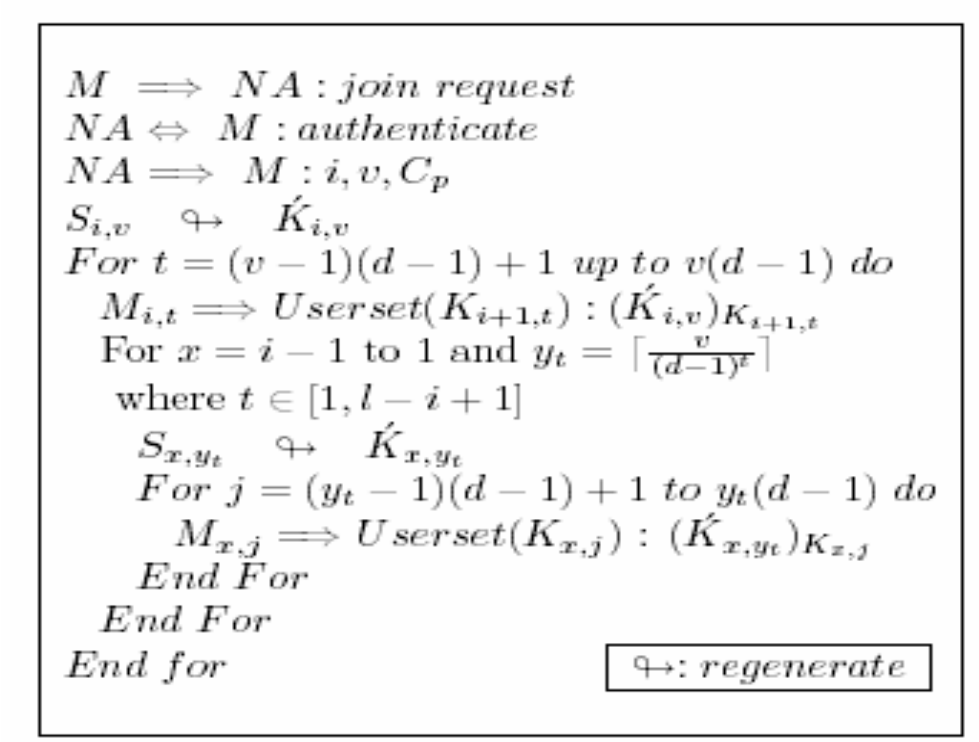

Figure 4: Join protocol

4.3.2 Partition Protocol

The partition operation removes *m* members from the current group of *n* numbers. If the partition occurs due to a network failure, there are two cases: 1) if the removed members belong to the same subgroup, or in other words these members are children of a same parent, the *NA* will deal with this situation as just one member (the parent) leaves (If this

parent will leave also). If the parent still in the group it will regenerate a new key as if this parent will join the group. 2) If these *m* members do not belong to the same parent, the *NA* can determine weather the remaining *n ¡ m* members need to construct a new hierarchy or keep the same hierarchy with regeneration of subgroup session keys of the remaining subgroups with the needed modification to the members' indexing. If the partition is due to multiple leave requests, it will perform the (previously mentioned) leave protocol *m*-times. As shown from the previous discussion, the clustering approach is more efficient in the case of partition, since the leaving members (with high probability) belong to the same parent.

4.3.3 Merge Protocol

The merge operation is used to add $m > 1$ members to the current group of n members. If $m < d$ it can be treated as m join protocols from the point of view of the subgroup where the m members will join (Note that using GDH.2 [6] the upflow rounds will be repeated m-times to update the intermediate date and the last round, wherein the new member $d+m$ will be the last member in the subgroup, performs just once). But with respect to the other subgroups it will be considered as one member join. If $m \geq d$, the m members can construct a new subgroup(
s) with their subgroup(s) session key(s) and one member can join an old subgroup. So with respect of the old n members, it will be considered as one member request to join the current group. The indexing of the $n+m$ members need to be updated by the NA.

4.3.4 Key Refresh Protocol

The key refresh protocol updates the group session key or the different subgroups may need to update their session keys. This operation should perform the same procedure as the leave protocol does when $m = 0$. In this case, the subgroup will regenerate its new subgroup session key and then broadcasts it encrypted with its lower level subgroups session keys.

5. Complexity analysis

This section analyzes the communication and computation costs for the aforementioned protocols, namely, initial key generation, join, leave, merge, and partition protocols. In our analysis, we consider the following costs:

Number of Rounds (*R*): Where the number of rounds is obviously an abstract measure of the time needed to perform the protocol ([20] p.133–134). On other words, the total number of rounds is the number of actions among members that need to be done serially. During a round, each member takes the following actions in the specified order:

1. It sends a message to its (lower level subgroup) neighbor.
2. It receives the message sent to it by its (parent) neighbor.
3. It changes the values of the message it has received by adding its exponent (or decrypt the message to get its upper level subgroup key).

Number of Messages (*M*): The total number of messages plus total number of broadcasts sent according to the protocol. Where a message is a single packet sent from one member to another. While a broadcast is a message sent by a member and received by a subset (or all) of the group members. So the number of messages measures the number of packets sent (where broadcast message considered as one message).

Number of Exponentiations (E_{xT}): Since the Modular Exponentiation (Given (α, *x*, *p*), find $y = \alpha^x \bmod p$) is the most expensive operation (it requires $O(log^3 p)$ bit operation in Z_p^*), the major concern in reducing the computation overhead is to reduce the total number of exponentiation need to be executed by the members through the protocol.

Maximum Bandwidth (BW_M): The maximum bandwidth is considered to be the size of the largest message sent during the protocol processing.

We compare our protocol with the authenticated contributory group key agreement scheme GDH.2 described in [7, 12]. To the best of our knowledge, GDH.2 of the Cliques [12] is the best protocol that provides contributory authenticated group key agreement, capable of dynamic membership events, and handling both partitions and merges. That is why we chose GDH.2 to be our building block of our protocol. Also we compare our protocol with TGDH [19]. TGDH is a new group key agreement protocol based on Diffie-Hellman key trees, which is the first proposal to consider the scalability problem. The computational complexity is reduced from *O* (*n*) to *O* (*log n*), but the communication cost
is slightly larger than that of GDH.2.

The overhead of our protocol depends on the structure of the hierarchy, i.e., the maximum number of the members in subgroups *d* and the maximum number of levels *l* in the hierarchy. As we saw from the characteristics of each protocol, there is a tradeoff between *d* and *l*. The number of rounds in the initial protocol requires *O* (*ld*) to generate the session key, in case of membership change protocols, on the other hand, it requires *O* (*l*) to regenerate the session key. The number of messages and the computation overhead (exponentiations and encryptions) on the initial protocols depends on the maximum number of subgroups, N_S (which is approximately *O* ($(d)^{l-1}$), so it will be affected more by increasing the number of levels. In the case of membership change protocols, the number of messages and the computation overheads requires *O* (*ld*) to regenerate the session key. To conclude, the number of levels has to be minimized to achieve the best efficiency, but at the same time we need to keep the balance of the hierarchy. Also the maximum number of

members in the subgroup has to be bounded above not so as to lose the benefits from our protocol. Compared with GDH.2, it is clear that the computation overhead in our protocols is superior, on the average it requires $O(ld)$ to generate/regenerate the session key compared with $O(n)$ (approximately $O(d^l)$) in the case of GDH.2. With respect to the communication overhead, the number of rounds in our initial key agreement protocol is lower than that of GDH.2. The number of messages, on the other hand is greater than that of GDH.2, but the maximum bandwidth (the size of the largest message sent for the operation) in our protocol is d compared with n in GDH.2, which allow parallel handling of the messages through the network. As a result the total load on the network from our protocol is less than that of GDH.2. Compared with TGDH, the communication overhead is slightly larger than that of GDH.2, so with respect to the communication overhead our protocol is superior. The computation overhead is approximately the same in both protocols, especially in case of membership change protocols, since both protocols are multi-round protocols. But our approach is more efficient for large size groups. Since the tree in TGDH is binary, the depth of the tree h ($h = log\ n$ if the tree is full balanced) is larger than the number of levels l of our protocol which reduces the number of rounds and exponentiations. Also the number of users involved in the key regeneration is lower in our protocol. Another interesting point in our protocol is the number of keys held by any member is lower than that of the TGDH, which reduces the required storage capability of each user. Finally, we have to mention that in our protocol we need to encrypt all the down messages from the upper levels to the lower ones to broadcast the upper level subgroups' keys, which is not required in GDH.2 or TGDH. The cost of the symmetric encryptions process is lower than the exponentiations used in key agreement protocol using public key. A summary of the comparisons are in Tables 1, 2.

Operation	Initial Key Generation		
Protocol	GDH.2	TGDH	CGDH
R	n	$2h$	$l(d+1)-1$
M	n	$2nh$	$d(N_S+1)$
Ex_T	$\frac{n(n+3)}{2}-1$	$2nh$	$E_{x_s}(N_S)$
BW_M	$n(n-1)$	$2n$	d

Table 1: Initial Key Generation Cost

Operation	Join protocol		
Protocol	GDH.2	TGDH	CGDH
R	2	6	$2l$
M	2	2	$2+d(l-1)$
Ex_T	$3n+2$	$\frac{3h}{2}$	$d+3+l(2d-1)$
BW_M	n	$2n$	d

Table 2: Join protocol Cost

6. Conclusion and Future Work

In this paper we presented a new framework for a scalable key agreement protocol. The most important feature of our protocol is the adoption of the Clustering in key agreement protocols. Using clustering, we are able to reduce the cost of session key generation/regeneration. The idea of clustering was first proposed in [15] to create hierarchies for wireless networks and was used in [14] as a scalable and secure distribution of the group (session) key. Our approach is based on an already existing key agreement protocol (GDH.2) as a building block to enable concurrently, and hence reduce the time needed to generate the group session key. As we mentioned before our protocol can be applied to any efficient key agreement protocol. In our protocol, we tried to consider the scalability problem in securing communication groups (multicasting communication) where many other proposals have been published (see for example [3], [9] [18], [15]). None of these protocols has considered the scalability of key agreement protocol, rather they have considered the scalability of key distribution protocols. Despite the aforementioned reasons (section 1) which motivated us to focus on the key agreement protocol, our protocol is approximately as efficient as the most efficient key distribution protocol of these protocols, but regarding the storage capability of the members, our protocol is superior. Some considerations deserve further study. First and foremost, although we proved (heuristically) that our protocol is efficient with large numbers of group members, there are many practical issues that have to be discussed. We claim in our protocol that all the subgroups at the same level generate their session key at the same time, which needs some sort of synchronization between subgroups. Also, if one member has been delayed due to any problem it must not affect the completion of the protocol. An efficient implementation of our protocol is required to consider these (and more) practical issues. Second, our protocol needs to be securely proved. The building block security proof is based on the intractability of the DH problem (see for example [19], [20]). In our protocol is a combination between key agreement and key distribution (actually, we consider our work as the midway between key distribution and key agreement). Also the number of keys which are held by any member is larger than that of GDH.2. As consequence the security proof of our protocol has to consider the characteristics of both and we claim that it is harder than that of GDH.2. Also information theoretic analysis of our protocol is needed regarding the probability of member deletion, partition and merge based on the number of keys held by each member. Similar work was done by Poovenrdan, and Baras in [13], but they consider the key distribution using key

Graph proposed independently in [2] and [5]. Moreover they consider only the case of one member revocation. We plan to prove the security of our protocol by defining a security model of the scalable key agreement problem for a certain scenario (many-to-many communication in a large size group with a dynamic membership) and to discus the different membership changes cases with different probabilities. Third, we plan to investigate fast algorithms for key agreement protocols, we are particularly interested in approaches that lead to reduced key length and/or reduced computation overhead, such as methods using elliptic curves. Namely, we are working now in extending the Tripartite Authenticated Key agreement using Pairings, which was proposed by Joux [1] and then modified by Sattam, and Kenneth [16]. We aim to scale their protocol using our hierarchy with the number of members in the subgroup limited to 3 to scale to a large size group, rather than just a group with three parties.

References

[1] A. Joux. *" A One Round Protocol for Tripartite Diffie-Hellman,"* In W. Bosma, editor, Proceedings of Algorithmic Number Theory Symposium - ANTS IV, Vol. 1838 of LNCS, pp. 385–394. Springer-Verlag, 2000.

[2] C. K. Wong, M. Gouda, S.S. lam, "*Secure Group Communications Using Key Graphs*", Proc. IEEE INFOCOM '99, Vol.2, pp.708–716, 1999.

[3] D. Steer, L. Strawczynski, W. Di±e, and M. Wiener, *"A Secure Audio Teleconference System, "* in Advances in Cryptology - CRYPTO' 88 No. 403 in LNCS, pp. 520–528, Springer-Verlag, 1990.

[4] D. M. Wallner, E. J. Harder, and R. C. Agee,"*Key Management for Multicast: Issues and Architectures,"* Informational RFC-2627, June 1999

Problems," In Mullender, S. (2*nd* ed.), *Distributed Systems*. NY: ACM Press, 1993.

[5] D. M. Wallner, E. J. Harder, and R. C. Agee, *"Key management for multicast: issues and archetictures,"* Information RFC, July 1997

[6] G. Ateniese, M. Steiner, and G. Tsudik, "*New Multiparty Authentication Services and Key Agreement Protocols*," IEEE JSAC, Vol. 18, NO. 4, April 2000, PP. 628–638.

[7] G. V. Chockler, I. Keidar, and R. vitenberg, "*Group Communication Specifications: A Comprehensive Study*," ACM Computing Surveys Vol. 33, No. 4, Dec. 2001, PP. 1–43.

[8] I. Ingemarsson, D. Tang, and C.Wong, "*A conference key distribution systems*," IEEE

Transactions on Information Theory, Vol. 28, No. 5, PP. 714–720, Sept. 1982

[9] Mittra S.,"*Iolus: A Framework for Scalable Secure Multicasting*," Proc. of the ACM IGCOMM '97, 1997.

[10] M. Burmester and Y. Desmedt, *"A secure and efficient conference key distribution system,"* in advances in Cryptology–EUROCCRYTP'94 1995, No. 950 in LNCS, Springer-Verlag.

[11] M. Steiner, G. Tsudik, and M. Waidner, *"Cliques: A new approach to group key agreement,"* IEEE ICDCS'97, May 1997.

[12] R. Canetti, J. Garay, G. Itkis, D. Micciano, M. Naor, and B. Pinkas, *"Multicast Security: A Taxonomy and Efficient Construction*", IEE/ACM Transactions on networking, 2000, Vol. 8, No. 1, PP. 16–30.

[13] R. Poovenrdan, and J. S. Baras, *"An Information theoretic Approach to multicast Key Management ,"* in Proc. of IEEE Information theory and Networking Workshop, Metsovo, Greece, June, 1999.

[14] S. Banerjee, and B. Bhattcharjee, *"Scalable Secure Group Communication over IP Multicast,"* JSAC Special Issue On Network Support for Group Communication 2002

[15] S. Banerjee, and S. Khuller *"A Clustering Scheme for Hierarichal Control in Wirless Networks,"* IEEE Infocom 2001, Anchorage, Alaska, April 2001.

[16] S. S. Al-Riyami, and K. G. Paterson *"Authenticated Three Party Key Agreement Protocols from Pairings,"* Information Security Group, Technical Report, March 20, 2002

[17] W. Diffie and M. Hellman, *"New direction in cryptography,"* IEEE Transactions on Information Theory, Vol. IT-22, PP. 644–654, Nov. 1976.

[18] Y. Kim, A. Perrig, and G. Tsudik, *"Simple and Fult-Tolerant Key Agreement for Dynamic Collaborative Groups,"* ACM CCS 2000, November 2000

[19] E. Bresson, O. Chevassut, D. Pointcheval, and Jean-Jacques *"Provably authenticated Group Diffie-Hellman Kay exchange,"* In Proc. of the 8*th* ACM CCS PP. 255–264. ACM Press, Nov. 2001.

[20] E. Bresson, O. Chevassut and D. Pointcheval *"Provably authenticated Group Diffie-Hellman Kay exchange - The Dynamic Case,"* In Proc. of Asiacrypt, volume 2248 of LNCS, PP. 290– 309. Springer Verlag, Dec 2001.