Queue-based Group Key Agreement Protocol

Sunghyuck Hong
Office of International Affairs, Texas Tech University
601 Indiana, Lubbock, Texas, USA
(Email: sunghyuck.hong@ttu.edu)
(Received Oct. 1, 2007; revised and accepted Aug. 22, 2008)

Abstract

Group communication is exploding in Internet applications such as video conferences, online chatting programs, games, and gambling. Since most group communication takes place over the Internet that is a wide open network, security plays a major role. For a secure communication, the integrity of messages, member authentication, and confidentiality must be provided among group members. To maintain message integrity, all group members use a Group Key (GK) for encrypting and decrypting messages during group communication. Secure and efficient group key managements have been developed to generate a GK efficiently. Tree-based Group Diffie-Hellman (TGDH) is an efficient group key agreement protocol to generate the GK. TGDH and other group key generation protocols assume that all members have an equal computing power. However, one of the characteristics of a distributed computing environment is heterogeneity; the member can be at a workstation, a laptop, or even a mobile computer. TGDH and other group key generation protocols assume all members have an equal computing power. However, one of the characteristics of distributed computing is heterogeneity. Therefore, this research considers member's diversity and proposes enhanced group key generation protocol with filtering out low performance members in group key generating processes to improve the efficiency of GK processes.

Keywords: Group key management, network security, protocol design, secure group communication

1 Introduction

Most group communications are occurred over the Internet in the form of video conferences, on-line chatting programs, games, and gambling. Security is a major concern in group communication. According to [19], member authentication processes and key distribution take place at the beginning of a group communication for a secure group communication. For establishing a secure group communication, all members contribute to generate and use a common group key. A Group Key (GK) plays a major security role. Since a Group Key Generation Process (GKGP) has many modular exponentiations, it takes a long time to compute a GK even though group size is relatively small as 100 [17]. For achieving a high level of security, the GK should be changed after every member joins and leaves so that a former group member has no access to current communications and a new member has no access to previous communications [19]. Therefore, group key management protocols have been focusing on generating a GK efficiently [12, 14, 17, 19]. In order to compute the GK using modular exponentiations, the adaptation of key trees is an efficient way to reduce the computational overhead. Modular exponentiation is the computationally most expensive operation in the group key management protocols [14]. The number of exponentiations for membership events depends on a group size. TGDH is the most efficient group key agreement protocol. The algorithm efficiency of TGDH is $O(log_2 n)$, where n is a group size, so it is efficient as long as the key tree is perfectly balanced. However, maintaining a perfect key tree balance causes another overhead. Furthermore, TGDH and other key management protocols assume that all members' machines are under a homogeneous computing and network environment. In general, members can be widely distributed over the Internet and can be at a workstation, a laptop, or even a mobile computer. For example, if a member using a mobile computer joins in group communication, then it takes more computational times to generate a GK than any other members machines. In addition, a member who is physically far away from other members locations or in a high network latency area will degrade the overall GKGP because multiple message exchanges are required while a GK is being generated. Therefore, a low performance member with high network latency must be filtered out for the efficient GKGP. In this research, Queue-based Diffie-Hellman group key agreement protocol is proposed for improving the diversity of computing environments and network latency. In [10], the tree-based group key agreement filtering out low performance members was proposed and proved the development of the efficiency for a GKGP. However, it still has a maintenance problem to make the tree balanced after membership changes. The proposed protocol provides a Queue-based divide-conquer algorithm with a low maintenance. A Queue structure is being used for determining

fast and low performance members. The tree-based group key agreement with filtering out low performance members [10] authorized fast members in group key generation processes by comparing each members elapsed times to compute their public keys, whereas in the proposed protocol, instead of comparing the elapsed times, a Group Controller Server (GCS) requires all members to computer their public keys and store they keys into the Queue structure on the GCS with the order of arrival. A Queue structure is First In First Out (FIFO). The fastest members key is always stored into the right most position in the Queue, the second fastest key is stored into the second right most position, and the third fastest key is stored so on. If using a Queue structure, then it determines fast members ease without any additional overhead such as making a tree balanced. The detail processes in the proposed protocol are dealt with in Section 2. A secure and efficient group key management is a critical issue in group communication [3]. For a secure and group efficient key management, it is necessary to filter out low performance members with regard to low computing power and high network latency. Therefore, this research focuses on increasing the efficiency of the GKGP which is aimed at maximizing group communications usability.

1.1 Related Work

Group communication occurs in many different settings: from low-level network multicasting to conferencing and other groupware applications. Regardless of the environment, security services are necessary to provide communication privacy and integrity. These are not possible without secure efficient key distribution, authentication, and other security mechanisms. For a secure communication, a group key management is responsible for generating the GK and distributing it to each member securely over an insecure network environment. Therefore, the key management is a building block in group key management. Unless the communication channel is secure, delivery of messages over the network to the right destination can not be guaranteed. There are basically two approaches for group key management in peer-to-peer group communications. Centralized group key distribution relies on a single entity (called a key server) to generate and distribute a GK to all group members. A Trusted Third Party (TTP) can make this approach possible. However, there are two problems: TTP must be constantly available, and TTP must exist in every possible subset of a group in order to support continued operation in the event of network partitions. The first problem can be addressed with fault-tolerance and replication techniques [14]. The second is impossible to solve in a scalable and efficient manner. In one-to-many multicast scenario the centralized approach works well because a TTP placed at, or very near, the source of communication can support continued operation within an arbitrary partition as long as it includes the source. In general, one-to-many settings only focus on offering continued operation within a single partition including the source. However, a dynamic peer-to-peer communication must offer continued operation in an arbitrary number of partitions. Therefore, the centralized approach to group key management is not well suited for a dynamic peer-to-peer group communication [14]. The decentralized group key distribution, which is called a group key agreement, involves dynamically generating a GK and distributing it to other group members. This approach is more robust and more applicable in dynamic peer-to-peer group communications. In contrast to the centralized approaches, contributory group key management requires each group member to contribute an equal share to the group key. This avoids the problems with the centralized trust and the single point of failure. The group key agreement in which each member has an equal opportunity for generating a GK is better suited for peer-to-peer group communication [17]. Therefore, this research is directly related with contributory and decentralized group key management. Currently, there are five different group key agreement protocols: BD (Burmester-Desmedt) [6], GDH (Group Diffie-Hellman) [17], CKD (Centralized Key Distribution) [2], STR (Skinny Tree) [19], and TGDH (Tree-based Group Diffie-Hellman) [14].

1) The BD (Burmester-Desmedt) protocol supports dynamic group operations. The protocol has a relatively low computational overhead due to two modular exponentiations. However, it needs more message exchanges to generate GK. The $GK = g^{K_1K_2+K_2K_3+...+K_{n-1}K_n}$

2) GDH (Group Diffie-Hellman) provides high security assurance. However, it has a relatively high computation cost ($O(n)$ modular exponentiation) and is relatively hard to provide robustness. The $GK = g^{K_1K_2K_3K_4...K_{n-1}K_n}$

3) CKD (Centralized Key Distribution) provides the same security guarantee as GDH. The only difference between GDH and CKD is a centralized key distribution center that generates the GK and distributes it to each member. The $GK = g^{K_1K_2K_3K_4...K_{n-1}K_n}$

4) The STR (Skinny Tree) protocol is modified to provide dynamic group operations. The protocol has a relatively low communication overhead and is well-suited for adding new group members. Robustness is easily provided. However, it is relatively difficult for member exclusion ($O(n)$ modular exponentiation) The $GK = g^{K_n g^{K_{n-1}...g^{K_2 g^{K_1}}}}$

5) TGDH (Tree-based Group Diffie-Hellman) provides the most efficient group key agreement protocol. The protocol has low communication overhead and low computation overhead ($O(logn)$ modular exponentiation). In addition, TGDH provides robustness. The $GK = g^{g^{g^{K_1K_2}g^{K_3K_4}}...g^{g^{K_{n-3}K_{n-2}}g^{K_{n-1}K_n}}}$

1.2 Tree-based Group Diffie-Hellman

The Group Diffie-Hellman (GDH) key agreement protocol [11] is an extension of the Diffie-Hellman (DH) key exchange protocol [7]. The GK computation is an important component of group key management in securing group communication; several efforts to enhance the group key generating protocol have been reported [6, 8, 14, 17, 19] in which every member must contribute in the computation of the GK. Therefore, group key management focuses on minimizing computational overhead due to its inherent expensive cryptographic operations [19]. Because of the complexity of the GK computation, the group key management adapts a key tree structure that reduces computational times. Key trees have been suggested in the past for centralized group key distribution systems to reduce the complexity of the key generation [2]. One such GKGP is the Tree-based Group Diffie-Hellman TGDH [10]. An example of the key tree-based group key generation protocol follows. In the binary key tree for generating a GK in Figure 1, each node $< l, v >$ represents a v-th node at level l in the tree and node $< l, v >'$s secret (private) key $K_{<l,v>}$ and a blind (public) key $BK_{<l,v>} = f(K_{<l,v>}) = g^{K<l,v>} mod\ p$, where g and p are 1,024 bit long integers. Every member holds the secret key along the key path. For simplicity, assume each member knows the blind keys in the key tree. The key paths are the shadowed nodes (node $< 0,0 >, < 1,0 >, and < 2,0 >$) in Figure 1. The final group key $K < 0,0 >$ in Figure 1 is computed with the key paths using blind keys $BK_{<3,0>}$, $BK_{<3,1>}$, $BK_{<2,1>}$, $BK_{<2,2>}$, $BK_{<3,6>}$, and $BK_{<3,7>}$ [3]. Therefore, the final group key can be computed as Equation (1):

$$K_{<0,0>} = g^{g^{g^{K_{<3,0>}} K_{<3,1>}\, g^{K_{<2,1>}}}\, g^{g^{K_{<2,2>}} g^{K_{<3,2>}} K_{<3,3>}}} \quad (1)$$

The TGDH has two major disadvantages. First, maintaining a balanced group key generation tree causes another overhead. The group key generation tree must be balanced at any given time so that the efficiency of group key computation would be $O(log_2 n)$. Otherwise, the performance of group key generation key would be worse than $O(log_2 n)$. The second is derived from no regard for member's diversity in that if a low performance member such as a mobile computer joins the GKGP, then it would take a longer time than other members at desktop computers. Therefore, we propose a new protocol to provide possible solutions for current problems.

1.3 Problem Definition

As mentioned earlier, members in a distributed computing environment can be distributed over the Internet. Furthermore, the members can be at a workstation, a laptop, or even at a mobile computer. Conventional group key agreement protocols request all members to contribute to a GK. Consequently, a low performance member or a member located on a high network latency area could make the process slow. To improve the GKGP, relatively high performance members should be authorized to participate in the generation of the GK and then distribute it to low performance members, which are not authorized to participate in the GKGP. Since the group key generation time depends on members computing power and network latency. Therefore, it is necessary to filter out low performance members with regard to their computing powers and network latency for avoiding unnecessary delay in GKGP.

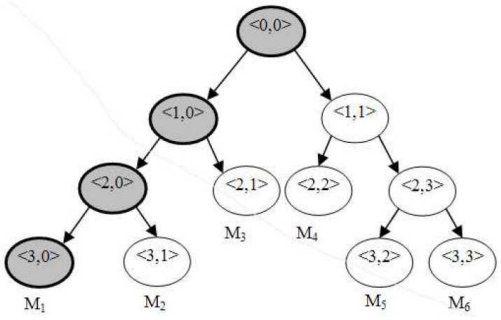

Figure 1: A binary tree for generating group key

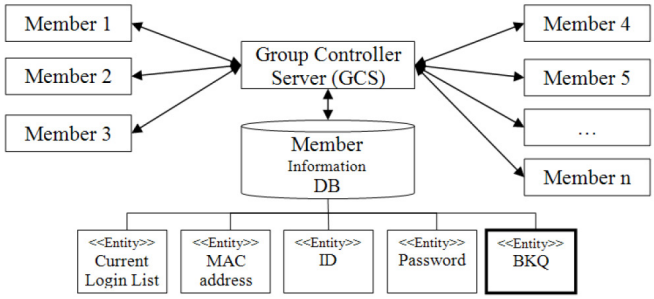

Figure 2: Queue-based group Diffie-Hellman entity model

2 Queue-based Group Diffie-Hellman (QGDH)

Figure 2 shows Queue-based Group Diffie-Hellman Entity Model. In Figure 2, a Group Controller Server (GCS) has a member information DB containing a current login member list, MAC addresses, IDs, Passwords, and Blind Key Queues (BKQ). If each member logins to the GCS, then the GCS will validate his ID, password, and MAC (Media Access Control) address [9] by checking his member information DB. After approving members identification, all members start to generate a group key by sending his blind key to the GCS who collects all blind keys and store them into the Blind Key Queues (BKQ) in order of arrival. Then the GCS informs participants who will join the next level of the group key generation.

A group controller server (GCS) is similar to the Vir-

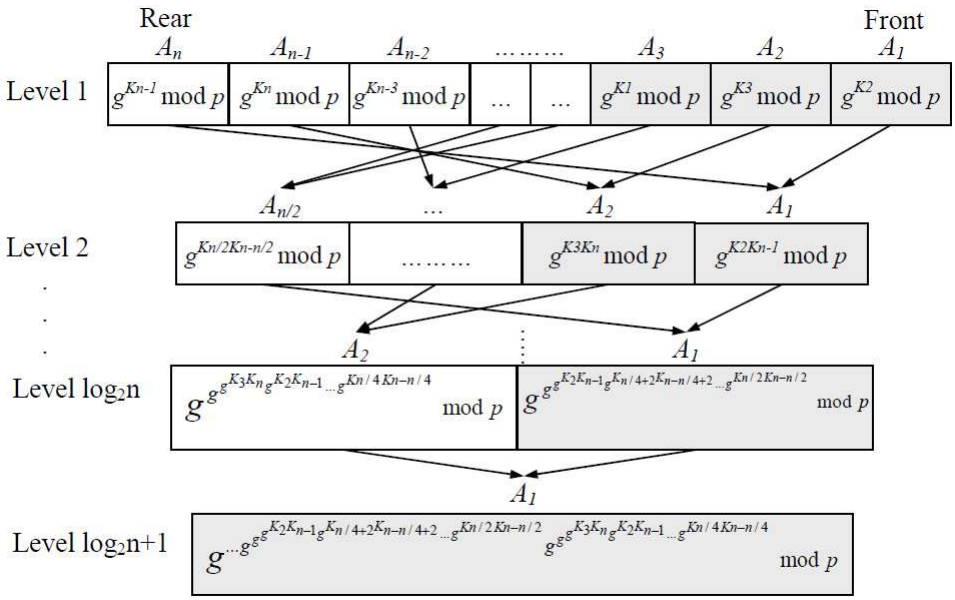

Figure 3: The blind key queues in group controller server

tual Synchrony [16] that uses a client daemon program to manage membership. The Virtual Synchrony (VS) daemon runs on each members machine and synchronizes with other VSs to update membership whenever membership changes. Each member can know other members status due to VS. TGDH uses a VS for maintaining memberships. However, Using VS involves complex processes to synchronize with all other VS daemons because members request a lot of message exchanges to update their status. In the mean time, the QGDH establishes a GCS server instead of running VS daemons on all members machines. Therefore, only one machine in the proposed protocol takes a responsible for a GCS.

Current group communication protocols use a self-signed certificate for member authentication, which has a well-known weakness in that members cannot ensure that the name on the public key is really a true member's name [8]. To compensate for this well-known weakness, our approach uses a GCS as a complete trusted party. Our threat model takes into account both passive and active outsiders (i.e. individuals who are not group members). Passive outsider attacks involve eavesdropping with the aim of discovering the GK(s); active outsider attacks involve injecting, deleting, delaying, and modifying protocol messages.

The proposed protocol authorizes only high performance members with low network latency join in the GKGP for maximum efficiency of the group key generation. A group key must be regenerated for secure communication whenever group membership changes. The details of the proposed procedure are described as follow.

Suppose that the group size is n and group members are $M_1, M_2, M_3, ..., M_{n-1}, M_n$ for n < 100. The Blind Key Queues (BKQ) in GCS are shown in Figure 3. The GCS requests all members to generate a blind key (g^{K_i} mod p, $i \in [1, n]$, g is an exponentiation base, p is a prime number, and K_i is i^{th} members private key) by broadcasting a request message. The GCS receives all blind keys and stores them into its own BKQ in the order of their arrival to determine high performance members joining in the next level of the GKGP. The highest performance members blind key is always stored in the front of the BKQ, whereas the low performance members blind keys go in the rear of the BKQ. In fact, two factors must be considered to determine high performance members. One is to measure members computing power and the other is to measure network latency. These two factors directly affect the performance of a GKGP. No matter how high performance the machine, if the time taken for communication messages to traverse the network is long, then the member must be regarded as low performance. Therefore, BKQ is a simplest way to measure for computing power and network latency at once. The number of levels in the GKGP can be determined by a group size. If the group size is n, then the number of levels is $log_2(n+1)$. In the first level, all members are required to generate a blind key. In the next level, The GCS assigns two blind keys located on the opposite side of the BKQ in Figure 3, and then the GCS requests members who are in the shaded spots in the BKQ to compute Diffie-Hellman key exchange with those blind keys and to store them into the BKQ in order of arrival. Following each level of group key computation, the GCS collects and stores all computed partial group keys into the BKQ in the order of their arrival at every level of the GKGP. As noticed in Figure 3, the fastest members key is always stored into the A_1 spot in each level, the second fastest members key is stored into the A_2 spot, and so on. The BKQ automatically assigns a pair of key computations. For example, if a member is in the A_1 spot, then he is assigned to compute with a

member in the last spot, An. Thus, A_1 spots blind key (g^{K_2} *mod p*) will be computed with A_n spots blind key ($g^{K_{n-1}}$ *mod p*), A_2 spots blind key (g^{K_3} *mod p*) and so on. After having completed all levels, the final group key can be computed as Equation (2):

$$g^{\cdots g^{g^{K_3 K_{n-1}} g^{K_2 K_{n-1}} \cdots g^{g^{K_{(n/4)} K_{n-(n/4)}}}}} \quad (2)$$

In this case, only fast members (the shaded area in Figure 3) are authorized to participate in the next level in the group generation processes. Therefore, the proposed protocol can prevent unnecessary delays and improve efficiency.

3 Performance Analysis

In this section we analyzed the computation costs and network latency for join, leave, merge, and partition of the proposed protocol, Queue-based Group Diffie-Hellman (QGDH). In order to evaluate the performance of the proposed protocol, I measured the total elapsed time from the moment the group membership event happened until the time when the group key generation processes finished. I compared the total elapsed times for all other group key generation protocols in the same network environment to prove the efficiency of the proposed protocol. The performance analyses of group key generation are shown in Figures 4 thru 7 to compare the protocol efficiencies.

3.1 Membership Operations

The proposed group key agreement protocol needs to provide membership operations to cope with membership changes. The QGDH includes protocols in support of the following operations:

1) Join: a new member is added to the group communication.

2) Leave: a member is removed from the group communication.

3) Partition: a subset of members is split from the group communication.

4) Merge: a partitioned group is merged with the current group communication.

This performance analyses focus on the number of rounds, the total number of control messages, network overheads, and group key generation costs. The total cost is the sum of all participants' costs incurred by any participant in a given round or protocols. I compared QGDH to other group key agreement protocols including TGDH [14], STR [19], GDH [17], and BD [6]. The number of current group members, merging members, merging groups, and leaving members are denoted by: $n, m, k(m \geq k)$ *and* p, respectively in Table 1. The height of the key tree constructed by the TGDH protocol is h.

Table 1: Communication and computation costs summary

Protocol		Communication		Computation
		Rounds	Messages	Exponentiation
QGDH	Join	2	$2n-2$	$\frac{3(log_2 n)}{2}$
	Leave	1	$2n-2$	$\frac{3(log_2 n)}{2}$
	Partition	1	$2n-2$	$3(log_2 n)$
	Merge	2	$2n-2$	$\frac{3(log_2 n)}{2}$
TGDH	Join	2	3	$\frac{3h}{2}$
	Leave	1	1	$\frac{3h}{2}$
	Partition	$min(log_2 P, h)$	$2h$	$3h$
	Merge	$log_2 K + 1$	$2k$	$\frac{3h}{2}$
STR	Join	2	3	4
	Leave	1	1	$\frac{3n}{2}+2$
	Partition	1	1	$\frac{3n}{2}+2$
	Merge	2	$k+1$	$3m+1$
GDH	Join	4	$n+3$	$n+3$
	Leave	1	1	$n-1$
	Partition	1	1	$n-p$
	Merge	$m+3$	$n+2m+1$	$n+2m+1$
BD	Join	2	$2n+2$	3
	Leave	2	$2n-2$	3
	Partition	2	$2n-2p$	3
	Merge	2	$2n+2m$	3

Table 1 shows the communication and computation costs of these five protocols. The cost of the protocols, TGDH, STR, GDH, and BD are reported in Table 1 [8]. The rounds in Table 1 illustrate how many broadcasts are used to send and receive blind keys among each member. In QGDH as a new member joins, the GCS (Group Controller Server) broadcasts a request to current members to generate a blind key. That is the first round. After sending the requesting message, the GCS collects and stores all other members' blink keys into the BKQ. Two rounds are needed in the join event. The first round is for broadcasting the requesting message to start generating a blind key. The second round is for receiving blind keys from all current members. Only one round is required in the leave event because the leaving member just notices his leaving to all others by using one broadcast. The total number of messages which is shown in Table 2 will be reduced to the half of messages in the previous level. According to [1], the infinite geometric series $\alpha + \alpha r + \alpha r^2 + ... \ \alpha r^n$ (n is infinite) is convergent and has the sum $\frac{\alpha}{(1-r)}$ if and only if -1 < r < 1. In this case, α (an initial value) $= n-1$, $r = \frac{1}{2}$. Thus, the total number of messages can be counted as the infinite geometric series. Finally, the sum of the messages is computed as $\frac{(n-1)}{(1-\frac{1}{2})}$ $= 2n-2$. In partition, the total number of messages is divided into m partitioned groups. Thus, the initial value a is $\frac{(n-1)}{m}$ and $r = \frac{1}{2}$. Even the initial value α is different from other events. However, the final total messages $\frac{\frac{(n-1)}{m}}{1-\frac{1}{2}}$ should be multiplied by m. so the total number of messages $(2n-2)$ are the same as in the other events.

Table 2: Total messages in join and leave for QGDH

GK Generation Levels	Total Number of Messages
Level 1	$n-1$
Level 2	$\frac{(n-1)}{2}$
Level 3	$\frac{(n-1)}{4}$
...	...
Level n	$\frac{(n-1)}{2^{(n-1)}}$
Total	$\sum$(n-1)$+\frac{(n-1)}{2}+\frac{(n-1)}{4}+\frac{(n-1)}{8}$ $+ \ldots + \frac{(n-1)}{2^{(n-1)}} = 2n-2$

Modular exponentiation is computationally the most expensive operation in TGDH, and STR [8]. The number of exponentiations for a membership event depends on group size. Therefore, there is a strong link between a computational overhead and group size in GKGP.

3.2 Test Method

For fair comparisons, I used g, k, and p = 1,024 bits long numbers for all measurements. These values are known to be secure in the current technology [15]. Also, I used the following scenario to measure delay. For join, leave, partition, and merge, the number of current group members is n (n = 10, 20, 30, 40, 50, *and* 60). The experimental test bed is a 64 Intel Pentium machine running Windows XP. The group members are uniformly distributed on a Local Area Network (LAN). One machine in the experimental test bed must take a role as a Group Controller Server (GCS) which is responsible for controlling the GKGP. However, GCS is not involved to compute a group key. Whenever membership changes, GCS requests all members to generate their blind keys and stores all the blind keys from the members into his/her own BKQ in the order of their arrival to determine fast members joining in the next level of the GKGP. Therefore, relatively low performance members are automatically filtered out at each level of the GKGP. For the performance test of the proposed protocol, the five group key generation protocols must be launched on the GCS. The elapsed times to compute a GK for each protocol have been measured on the GCS whenever membership changes during group communication. Each member on a single machine acts as a group member in the group key generation protocol. Tests performed at our test bed show that the average elapsed times for computing a GK and the network latency that included communication overheads by delivering messages among group members. I tested more than 30 times by increasing or decreasing one machine at a time. In LAN environment, communication costs are relatively small ranging from 17 μsec to 25 μsec for sending a 1,024-bits long message. It is negligible for a maximum group size 60. The large portion of the costs is a computational cost which is closely related with a group size. However, in WAN (Wide Area Network), the major overhead is completely the other way around. According to

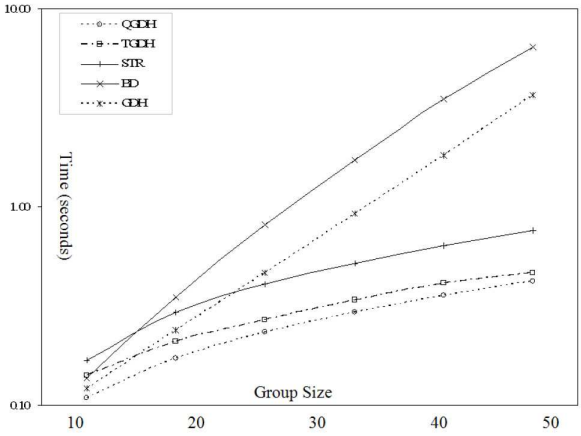

Figure 4: Join cost comparison

[13], the elapsed time from the USA to Thailand took 420 μsec and 670 μsec was measured from the USA to Mozambique. The communication costs in LAN are million times faster than in WAN. WAN (Wide Area Network)-based secure group communication will be dealt with by future research. To prove the efficiency of the proposed protocol, I demonstrated and showed the differentiation between the proposed protocol with filtering and other four non-filtering protocols by using plotting the elapsed times for each event.

3.3 Join Operation Results

The overheads are computation and communication costs. Figure 4 shows that BD and GDH are inefficient join cost in terms of group size. In Figures, the x-axis denotes the number of members and the y-axis denotes a second for computational cost. On the other hand, QGDH, TGDH, and STR are efficient because they use a divide-and-conquer algorithm to compute a GK. QGDH is most efficient, scaling logarithmic in the number of exponentiations. Both TGDH and STR use a binary tree to compute a GK. QGDH has $2n-2$ message communications that are more messages than any others. However, the relatively large message communications do not negatively affect the performance to compute a GK because it takes at most 20 μsec to exchange messages among group members in a GKGP. Therefore, QGDH is more efficient than other tree-based GK computation protocols.

3.4 Leave Operation Results

Figure 5 shows the computational delay for a random member to leave a group of n (n = 10, 20, 30, 40, 50, *and* 60). The leave cost depends on how many remain members are, so it does not matter how many members left. Thus, the leave cost is almost the same as the join cost. However, the computation cost for STR upon a leave event depends on the location of the leaving member on the key generation tree. Therefore, in leave cost for STR is not as good as join case.

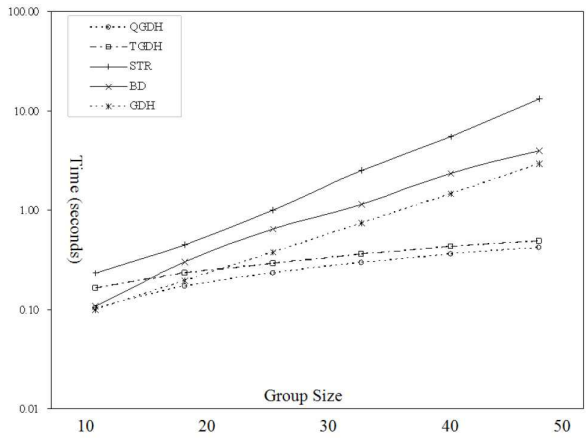

Figure 5: Leave cost comparison

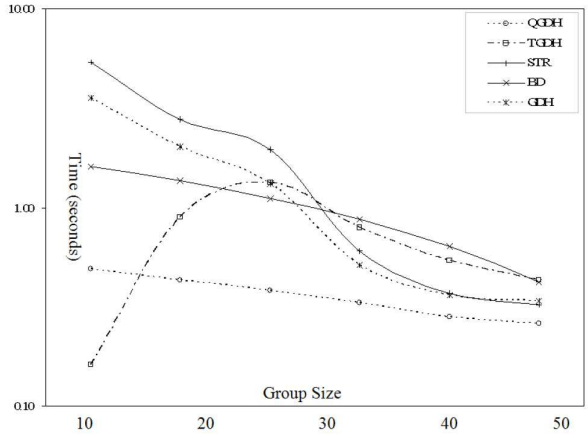

Figure 6: Partition cost comparison

3.5 Partition Operation Results

In Figure 6, x axis denotes the number of remaining members after partitioning the group. Whenever a network fault takes place, the result of a network fault may divide a group into several subgroups. I forced the group into several parts and get them recomputed their sub group keys. For BD and GDH, the location of the partitioning members does not matter. However, it is important in QGDH, STR and TGDH. If the network fault is detected in TGDH, each remaining member updates its tree by deleting all partitioned members as well as their respective parent nodes. Due to overheads in partition event, TGDH is not efficient as much as other events. However, STR, BD, GDH, and QGDH are almost same as the reverse of the leave event.

3.6 Merge Operation Results

Figure 7 shows merge operation results. After recovering network fault, merge operation will occur. The overall performances of all protocols are good except GDH. The performance of GDH strongly relays on the number of current group size. In addition, it has $n + 2m + 1$ exponentiations and $n + 2m + 1$ communications. The current

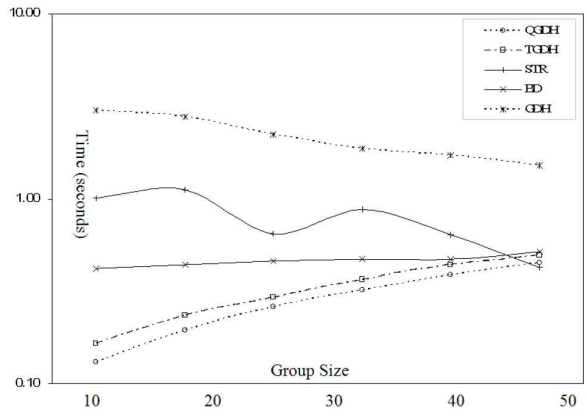

Figure 7: Merge cost comparison

group size is not important factor for QGDH and TGDH. QGDH and TGDH in the merge operation are almost same as the leave operations. Therefore, the divide and conquer based-based protocols' performance are almost constant.

Based on the experimental results on computational cost, QGDH shows the best performance despite relatively large message communications. QGDH adapts a Queue structure to generate a GK. In addition, QGDH considers user's diversity, so only approved members will be able to join in generating GK processes. Therefore, in distributed computing environment user's diversity must be considered to improving the efficiency of a group key generation.

4 Conclusions

Without employing strong security features, a group communication will not fully accomplish its goal of enabling communication between members who share common interests. A group communication should have strong security features to protect members' privacy and message integrity. A Group Key (GK) is in the heart of the security features in a secure group communication. To generate a GK takes a relatively long time for low performance machines. However, it must be accomplished for a secure communication no matter what machines are used. Therefore, the group key management must be developed to generate a GK in efficient and secure ways. Mobile computers have been becoming popular, and network clusters have been communicating with conventional servers in recent years. Besides, group members are widely distributed over the Internet and they can use any kinds of machine as long as connecting to the Internet. Therefore, a group key management must consider member's diversity and network environments because the characteristics of a distributed computing environment is heterogeneity, and most group members are widely distributed over the Internet. Based on these assumptions, I proposed and proved the efficiency of the Queue-based Group Diffie-Hellman protocol with compar-

ing to other protocols. The proposed protocol enhanced a group key generation processes by using a Queue structure with filtering out low performance members based on its computing power and network latency. Therefore, my research contributes to the achieving of maximum efficiency in GKGP and improving group communication's usability.

References

[1] M. Abramowitz, and I. A. Stegun, *Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables*, 9th printing, New York: Dover, p. 10, 1972.

[2] Y. Amir, G. Ateniese, D. Hasse, Y. Kim, C. N. Rotaru, T. Schlossnagle, J. Schultz, J. Stanton, and G. Tsudik, "Secure group communication in asynchronous networks with failures, Integration and experiments," *IEEE International Conference on Distributed Computing Systems*, pp. 330-343, 2000.

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

[4] Y. Amir, Y. Kim, and C. N. Rotaru, "On the performance of group key agreement protocols," *ACM Transactions on Information and System Security*, vol. 7, no. 3, pp. 457-488, 2004.

[5] E. Bresson, O. Chevassut, D. Pointcheval, amd J. Quisquater, "Provably authenticated group Diffie-Hellman key exchange," *Proceedings of the 8th ACM Conference on Computer and Communications Security*, pp. 255-264, Philadelphia, PA, 2001.

[6] M. Burmester, and Y. Desmedt, "A secure and efficient conference key distribution system," *Advances in Cryptology - Eurocrypt'94*, pp. 275-286, 1994.

[7] W. Diffie, and M. E. Hellman, "New directions in cryptography," *IEEE Transactions on Information Theory*, IT-vol. 22, no. 6, pp. 644-654. Nov. 1976.

[8] M. Fratto, "In PKI Ww Trust?," *Network Computing*, vol.12, no. 18, pp. 69-77, Sep., 2001.

[9] S. Hong, and N. L. Benitez, "Media access control (MAC) address-based group key authentication scheme," *The 9th World Multiconference on Systemics, Cybernetics and Informatics*, pp. 160-164, Orlando, Florida, USA, July 2005.

[10] S. Hong, and N. L. Benitez, "Enhanced Group Key Computation Protocol," *The 2006 International Conference on Security and Management (SAM'06)*, Las Vegas, USA, June 26-29, 2006.

[11] Y. Kim, A. Perrig, and G. Tsudik, "Simple and fault-tolerant key agreement for dynamic collaborative groups," *The 7th ACM Conference on Computer and Communications Security*, pp. 235-244, ACM Press, Athens, Greece, Nov. 2000.

[12] Y. Kim, A. Perrig, and G. Tsudik, "Communication-efficient group key agreement," *17th International Information Security Conference (IFIP SEC'01)*, pp. 229-244, June 2001.

[13] Y. Kim, *Group Key Agreement: Theory and Practice*, Ph.D. thesis, May 2002.

[14] Y. Kim, A. Perrig, and G. Tsudik, "Tree-based group key agreement," *ACM Transactions on Information and System Security*, pp. 60-96, 2004.

[15] A. K. Lenstra, and E. R. Verheul. "Selecting cryptographic key sizes," *99 PricewaterhouseCoopers CCE newsletter*, Nov. 1999. http://www.cryptosavvy.com/

[16] L. E. Moser, Y. Amir, P. M. M. Smith, and D. A. Agarwal, "Extended virtual synchrony," *Proceedings of the IEEE 14th International Conference on Distributed Computing Systems*, pp. 56-65, IEEE Computer Society Press, Los Alamitos, CA, 1994.

[17] M. Steiner, G. Tsudik and M. Waidner, "Key agreement in dnamic peer groups," *IEEE Transactions on Parallel and Distributed Systems*, vol. 11, no. 8, pp. 769-780, Aug. 2000.

[18] D. Wallner, E. Harder, and R. Agee, *Key Management for Multicast: Issues and Architecture*, Internet-Draft draft-wallner-keyarch-00.txt, June 1997.

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

Sunghyuck Hong received his B.A. degree from Myongji University, Korea in 1995. After graduation, he worked at Hyosung Inc. in Seoul, Korea from 1995 to 1999 as a computer programmer and ERP consultant. He has a Ph.D. degree from Texas Tech University in August, 2007 major in Computer Science. Currently, he works at International Affairs in Texas Tech University as a senior program/analyst, and he is a member of editorial board in the Journal of Korean Society for Internet Information (KSII). His current research interests include secure authentication, secure group communication, biometric authentication, and key management protocol. To God, Gloria In Excelsis Deo.