

PERFORMANCE OF GROUP KEY AGREEMENT PROTOCOLS OVER MULTIPLE OPERATIONS*

Shanyu Zheng and Jim Alves-Foss
Center for Secure and Dependable Systems
University of Idaho
Moscow, ID 83844, U.S.A.
email: [zhen8299,jimaf]@uidaho.edu

Stephen S. Lee
Department of Statistics
University of Idaho
Moscow, ID 83844, U.S.A.
email: stevel@uidaho.edu

ABSTRACT

A few group key protocols are analyzed, implemented and deployed, but the costs associated with them have been poorly understood. Their analysis of group key agreements performance is based on the cost of performing a single operation. In this paper we extend this analysis to examine the performance behavior of five group key protocols after execution of multiple operation. We report our experimental results for 100 operations consist of combinations of join, leave, mass join, mass leave, merge, and partition. In order to thoroughly compare the performance of five protocols, we simulate three group operations: join-leave-mass join-mass leave, merge-partition, and join-leave-mass join-mass leave-merge-partition to observe what is not apparent from the theoretical analysis.

KEY WORDS

group key management, group communications, cryptographic protocols

1 Introduction

With wide use of the Internet, applications such as conference calls, voice conferencing, distributed computation, white-boards and distributed databases would strongly benefit from an enhanced multi-party group key exchange algorithm. To ensure secure and reliable communication between multiple groups, several attempts have been made to create a computation-communication efficient group key protocol for large and dynamic groups [6, 10, 4, 5, 7, 11, 3]. Three categories (centralized, distributed, and contributory group key managements) have been proposed to solve group key establishment [12, 1, 2]. In this paper we focus on contributory group key management, in which each participant generates its own private key and corresponding public value, and equally contributes to the shared group key which is derived by all the group members collectively.

*This material is based on research sponsored by AFRL and DARPA under agreement number F30602-02-1-0178. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of AFRL and DARPA or the U.S. Government.

It alleviates the problem of a single point of failure and trust in a centralized group key management, and also solves long term channel maintenance problems in a distributed group key management. In this paper we analyze the performance of five notable group key protocols based on contributory group key management: GDH3.0 [12], EGK [1], TGDH [9], STR [8], CCEGK [14] which are based on two-party Diffie-Hellman Key exchange.

The remainder of this paper is organized as follows: in Section 2, we briefly discuss these five contributory group key agreements and supplement partition operation in TGDH and STR, concluding with a summary of the theoretical performance of the five protocols in worst-case scenario. In Section 3, we discuss the simulation setup, and test scenarios. In Section 4, we report our simulation results, and give the conclusion in Section 5.

2 Related Work

There have been several group key protocols: GDH3.0 [12], EGK [1], TGDH [9], and STR [8], CCEGK [14], introduced that are based on contributory group key management to reduce the costs of communication and computation until now. These protocols are based on the classic two-party Diffie-Hellman (DH) approach for key exchange, extending it to multiple parties.

In these group key systems, several group key management operations: initialization, join, mass join, leave, mass leave, merge, partition, and key refresh, are used. Communication costs include number of rounds, number of unicast messages, number of broadcast message, and number of messages. Computation costs consist of total sequential exponentiations, total signatures, and total verifications [2, 14].

GDH3.0 is based on a linked list structure, so the number of sequential key exchange operations performed scales linearly with group size. EGK, TGDH, STR, and CCEGK are based on binary tree structures to provide for an efficient number of message exchanges, but they provide different algorithms for the group key and group management protocols.

Since partition operation for TGDH and STR is not fully documented, their performance is based on our best guess of their implementation of their operations. We im-

plement the partition operation for TGDH and STR as follows: suppose the current group size n is split into K subgroups $\{G_1, \dots, G_K\}$. We treat it as K mass leave operations. Each subgroup executes a mass leave operation. Because each subgroup does the mass leave operation in parallel, the total rounds are the maximum rounds among K mass leave operations, the total messages are the maximum of the messages among K mass leave operations, and the total sequential exponentiations are the maximum sequential exponentiations among K mass leave operations. The mass leave operation for TGDH and STR is described in [9, 8]

2.1 Theoretical Comparison

Table 1 summarizes the costs of communication and computation in seven operations: initialization, join, mass join, leave, mass leave, merge, and partition among the five protocols: GDH3.0, EGK, TGDH, STR, and CCEGK by theoretical analysis in the worst case. Because not all operations are fully documented in the literature, the performance of initialization, partition operations are based on our best guess of their implementation.

We use n, K_a, K_p, K_m, m to denote the sizes of current group members, mass join members, mass leave members, partitioned groups, merging groups, merging members, respectively. h and h_i are the height of the updating key tree and the height of the key tree of the i_{th} subgroup, respectively. h'_a, h'_m are $\lceil \log_2 K_a \rceil$ and $\lceil \log_2 K_m \rceil$, respectively. P_i is the the total size of leaving members for subgroup G_i . ρ, μ, μ, φ and ω are calculated as $\min(\lceil \log_2 K_p \rceil + 1, h)$, $\min(2K_p, n - K_p)$, $\max(\min(\lceil \log_2 P_i \rceil + 1, h_i))$, $\max(\min(2P_i, n - P_i))$, respectively.

Table 1 lists the costs of the evaluation metrics among these five protocols in worst case scenario. It is important that we further see how these protocols operate when multiple operations are executed. In Section 3 we therefore compare the average costs of multiple instances of these operations to get a better feel by experimental simulation.

3 Simulation Setup

Amir. et. al. compare the performance of each single operation among GDH3.0, BD, CKD, TGDH, and STR for different group size [2]. Zheng. et. al. compare the communication and computation costs of each single operation among EGK, STR, TGDH and CCEGK for different group size [14]. The drawback of their simulations is that they only compared the costs of single operations and did not consider the costs of combined operations. For example, Amir et.al report that for the join results of their simulation, STR outperforms other protocols [2]. However, for the leave operation, TGDH outperforms other protocols. From these results, it is difficult to determine that TGDH is more efficient than STR, because we do not know what

happens if we run join and leave operations together.

In our simulation, we consider the average costs of combined operations among GDH3.0, EGK, TGDH, STR, and CCEGK. In our previous work [13], we have done two categories of group operations: **join-leave** and **mass join-mass leave**. In this paper, we further do other three group operations: **join-leave-mass join-mass leave**, **merge-partition**, and **join-leave-mass join-mass leave-merge-partition**.

Join-leave-mass join-mass leave operation is a combination of the four single operations: join, or leave, or mass join, or mass leave, is executed every time. In our simulations, we suppose the occurrence of join, leave, mass join, and mass leave operation is independent and multinomial distribution.

Merge-partition operation is that one of two operations: merge or partition, is run every time. In our experiments, we assume that the occurrence of merge and partition operations is independent and uniform distribution.

Join-leave-mass join-mass leave-merge-partition operation is a combination of the six operations: join, leave, mass join, mass leave, merge, and partition, is run every time. In our tests, we suppose that the occurrence of these six operations is independent and multinomial distribution.

3.1 Test Scenarios

The costs of communication and computation in each protocol depend on a number of factors. We design our simulations that we take into account some of these factors. For example, the costs of GDH3.0 do not depend on the position of the joining or leaving members; the costs of all leave, or all join operations are equal. For EGK, its communication and computation costs do not depend on the position(s) of the joining or leaving member(s) because in the join, or mass join, or merge operations, it adds the new member(s) or merging groups to the root of the original key tree, and in the initialization, leave, mass leave, and partitioned operations we reconstruct a new balance binary tree. The communication and computation costs of TGDH, CCEGK depend on: the position(s) of the leaving or joining member(s), tree height, and the balancedness of the tree. STR's communication and computation costs depend on the position(s) of leaving member(s), and the size of merging group members. Based on the above analysis, we restricted our simulations as follows:

- For GDH3.0, we use the initialization operation to create the original linked list, which simulates the structure of the list at any point in time.
- For EGK, TGDH, and CCEGK, we first build a random, unbalanced binary tree (not using their initialization operation) to create the original tree structure that could possibly exist at some point in time.

Protocols		Communication Costs				Computation Costs		
		Rounds	Messages	Unicast	Broadcast	Sequential Exponentiations	Signatures.	Verifications.
CCEGK	Initialize	h	2n-2	n	n-2	2h-2	h	2n-2
	Join	1	2	1	1	1	1	2
	Mass Join	1	$K_a + 1$	0	$K_a + 1$	K_a	1	$K_a + 1$
	Merge	1	K_m	0	K_m	$K_m - 2$	1	K_m
	Leave	1	1	0	1	3h-3	1	1
	Mass Leave	ρ	μ	0	μ	3h-3	ρ	μ
	Partition	φ	ω	0	ω	$\max(3h_i) - 3$	φ	ω
	Partition	h	2n-2	0	2n-2	2h-2	h	2n-2
EGK	Join	1	2	0	2	1	1	2
	Mass Join	h'+1	$2K_a$	0	$2K_a$	$2h'_a$	$h'_a + 1$	$2K_a$
	Merge	K_m	$2K_m - 2$	0	$2K_m - 2$	$2K_m$	K_m	$2K_m - 2$
	Leave	h	2(n-1)	0	2(n-1)	2h	h	2(n-1)
	Mass Leave	h	$2(n - K_p)$	0	$2(n - K_p)$	2h	h	$2(n - K_p)$
	Partition	$\max(h_i)$	$\max 2(n - P_i)$	0	$\max 2(n - P_i)$	$\max(2h_i)$	$\max(h_i)$	$\max 2(n - P_i)$
	Initialize	h	2n-2	0	2n-2	2h-2	h	2n-2
	Join	2	3	0	3	3h-3	2	3
TGDH	Mass Join	$h'_a + 1$	$2K_a$	0	$2K_a$	3h-3	$h'_a + 1$	$2K_a$
	Merge	$h'_m + 1$	$2K_m$	0	$2K_m$	3h-3	$h'_m + 1$	$2K_m$
	Leave	1	1	0	1	3h-3	1	1
	Mass Leave	ρ	μ	0	μ	3h-3	ρ	μ
	Partition	φ	ω	0	ω	$\max(3h_i) - 3$	φ	ω
	Initialize	n-1	2n-2	0	2n-2	2(n-1)	n-1	2n
	Join	2	3	0	3	4	2	3
	Mass Join	2	$K_a + 2$	0	$K_a + 2$	$3K_a + 1$	2	$K_a + 2$
STR	Merge	2	$K_m + 1$	0	$K_m + 1$	3m+1	2	$K_m + 1$
	Leave	1	1	0	1	$\frac{3n}{2} + 2$	1	1
	Mass Leave	1	1	0	1	$\frac{3n}{2} + 2$	1	1
	Partition	1	1	0	1	$\frac{3n}{2} + 2$	1	1
	Initialize	n+1	2n-1	2n-3	2	5n-6	n+1	2n-1
	Join	4	n+3	0	n+3	n+3	4	n+3
	Mass Join	$K_a + 3$	$n + 2K_a + 1$	0	$n + 2K_a + 1$	$n + 2K_a + 1$	$K_a + 3$	$n + 2K_a + 1$
	Merge	m+3	n+2m+1	0	n+2m+1	n+2m+1	m+3	n+2m+1
GDH3.0	Leave	1	1	0	1	n-1	1	1
	Mass Leave	1	K_p	0	K_p	$n - K_p$	1	1
	Partition	1	1	0	1	$\max(n - P_i)$	1	1

Table 1: Performance Metrics for Five Group Key Agreement Protocols

- For STR, we obey the specified rules to create the binary tree, using its initialization operation.

We initially conducted our experiments with initial group sizes 200, 600, and 1000. We saw that the relative performance of the protocols was the same in these groups, so we only demonstrate the results for groups of 600 in this paper. For the experiments of **join-leave-mass join-mass leave**, **join-leave-mass join-mass leave-merge-partition**, and **merge-partition** we ran 100, 100, 50 operations, respectively. The operations of **join-leave-mass join-mass leave** and **join-leave-mass join-mass leave-merge-partition** are chose to 9 scenarios and 10 scenarios, respectively. To summarize our experimental configuration was:

- total operations in **join-leave-mass join-mass leave** and **join-leave-mass join-mass leave-merge-partition**=100. Total operations in **merge-partition**=50.
- initial group size for every group operation=600.
- result presented are averages over 10 runs.
- during a partition operation, the original group is divided into two subgroups of equal group size. After doing a partition, we randomly choose one of partitioned subgroups as the current group.

- for leave, mass leave, partition experiments, the leaving member are chosen randomly over all members.
- for mass join or mass leave, randomly choose from 2 to 10 members to join or leave.
- in the **join-leave-mass join-mass leave**, the probabilities of occurrences of the join and leave operations are equal, 35%, followed by the probabilities of occurrences of the mass join and mass leave operations which are both 5%. In **merge-partition**, the probabilities of the occurrences of merge and partition operations are equal, 50%. In the **join-leave-mass join-mass leave-merge-partition** operation, the probabilities of occurrences of the join and leave operations are equal at 35%, the probabilities of occurrences of the mass join and mass leave operations which are equal at 10%, the probabilities of occurrences of the merge and partition operations which are equal at 5%.

In every above group operation, we compute the average phases, average messages, average sequential exponentiations, average signatures, and average verifications. Since average verifications are the same to average messages and average signatures are the same to average phases, we use the same graphs to describe them. If the average value of each evaluation metrics is much smaller, its cost is more efficient.

4 Results

4.1 Join-leave-mass join-mass leave Results

Scenario	Join	Leave	Mass join	Mass Leave
1	90	5	2	3
2	80	10	5	5
3	70	20	5	5
4	60	25	7	8
5	50	30	10	10
6	40	40	10	10
7	30	45	12	13
8	20	50	15	15
9	10	60	15	15

Table 2: Nine Scenarios in Join-LLeave-Mass join-Mass Leave

Table 2 shows 9 Scenarios used in our simulation to demonstrate the tendency of the performance among the five protocols in the join-leave-mass join-mass leave operation. From Scenario 1 through Scenario 9, join operation is decreasing and leave, mass join, mass leave operations are increasing.

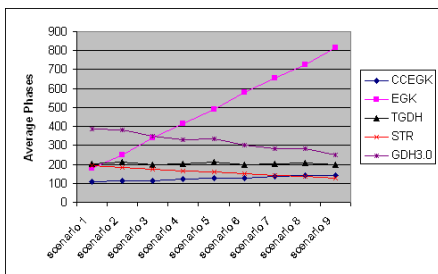


Figure 1: Join-LLeave-Mass join-Mass leave–Average Phases for five protocols

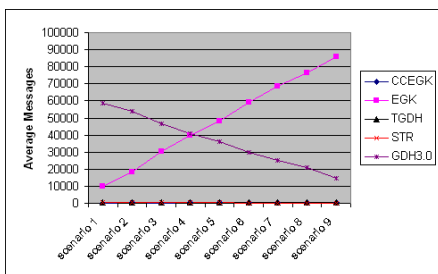


Figure 2: Join-LLeave-Mass join-Mass leave–Average Messages for five protocols

Figure 1 shows average phases, and Figures 2 and 3 demonstrate average messages, and Figure 4 presents average sequential exponentiations in the join-leave-mass join-mass leave operation when the number of every original group members is 600.

In Figure 1, when the total join operation is decreasing and total leave, mass join, mass leave operations

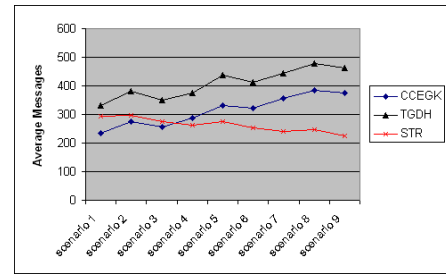


Figure 3: Join-LLeave-Mass join-Mass leave–Average Messages for the best three protocols

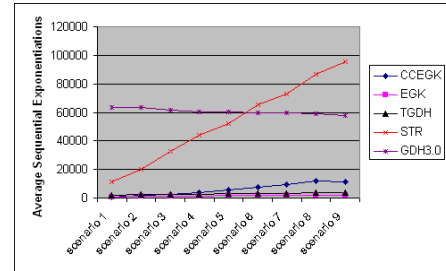


Figure 4: Join-LLeave-Mass join-Mass leave –Average Sequential Exponentiations for five protocols

are increasing, average phases of CCEGK are increasing steadily; however, average phases of TGDH, GDH3.0, and STR are decreasing steadily while average phases of EGK are rising rapidly. This is because CCEGK, TGDH, and EGK were designed to be constant for joins. CCEGK also keeps constant for mass joins, leaves, but scales with the height tree for mass leaves. TGDH scales with the joining number for mass add and with the height tree for mass leaves, but keeps constant for leaves. EGK scales with the joining members for mass joins but with the height tree for leaves and mass leaves. GDH3.0 keeps constant for joins, leaves, and mass leaves but scales with the joining members for the joining. For the long term, the average phases of CCEGK are smallest, followed by STR, TGDH, GDH3.0, EGK.

In Figures 2 and 3, when the total join operation is decreasing and total leave, mass join, mass leave operations are increasing, average messages of CCEGK, and TGDH are increasing steadily. However, the average messages of STR are decreasing steadily. Average messages of EGK are increasing sharply while average messages of GDH3.0 are dropping rapidly. This is because EGK was designed to be efficient for joins and mass joins but used a built in balancing mechanisms for leaves, and mass leaves, by reconstructing the tree after every leave or mass leave. STR keeps constant for every operation. GDH3.0 scales with the group size for joins and mass joins, and with the leaving members for mass leaves and keeps constant for leaves. TGDH and CCEGK were designed to keep constant for joins, leaves, but to scale with the joining members and

leaving members for mass joins and mass leaves. In the long term, the average messages of STR are smallest, followed by CCEGK, TGDH, GDH3.0, EGK.

In Figure 4, when the total join operations are decreasing, and total leave, mass join, mass leave operations are increasing, average sequential exponentiations of CCEGK, TGDH, and EGK are increasing steadily. However, average sequential exponentiations of STR are increasing quickly while average sequential exponentiations of GDH3.0 are falling steadily. This is because EGK rebuilds the tree after every leave or mass leaves, it scales with the height of key tree for leaves and mass leaves, but keeps constants for joins and scales with joining members for mass joins. GDH3.0 scales with group size for every operation and is the most expensive protocol. TGDH and CCEGK do a little change for updating the key tree. In every join and mass join operation, the key tree in CCEGK becomes more imbalanced than that in TGDH. In future we can use the rebalance schemes put forward in CCEGK to reduce its average sequential exponentiations. The key tree of STR becomes most imbalanced, so it almost scales with the group size. From the long term, the average sequential exponentiation of EGK are smallest, followed by TGDH, CCEGK, STR, GDH3.0.

4.2 Merge-Partition Results

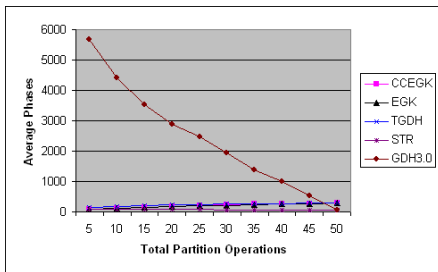


Figure 5: Merge-Partition-Average Phases for five protocols

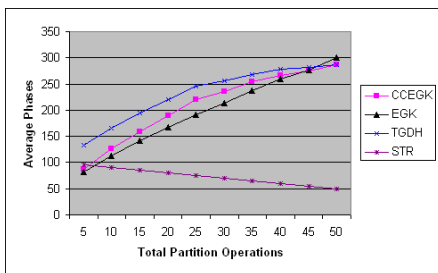


Figure 6: Merge-Partition-Average Phases for the best three protocols

Figures 5 and 6 show average phases, and Figure 7 demonstrates average messages, and Figures 8 and 9

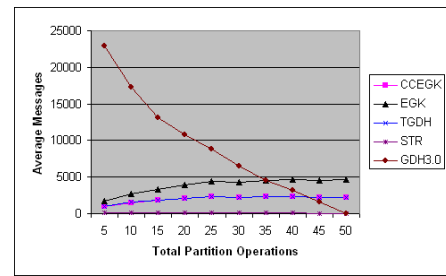


Figure 7: Merge-Partition-Average Messages for five protocols

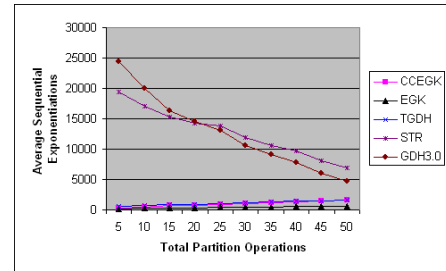


Figure 8: Merge-Partition-Average Sequential Exponentiations for five protocols

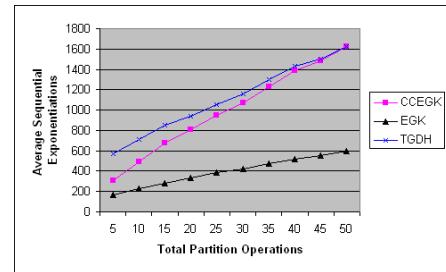


Figure 9: Merge-Partition operation-Average Sequential Exponentiations for the best three protocols

present average sequential exponentiations in the merge-partition operation when the number of each original group members is 100.

In Figures 5 and 6, when the total partition operations are increasing, average phases in STR are decreasing steadily. However, average phases of EGK, CCEGK, and TGDH are gradually rising while average phases of GDH3.0 are dropping quickly. This is because leaving group members is very large for every subgroup, it is almost equal for partition among EGK, CCEGK and TGDH. STR requires 2 phases for a merge and 1 phase for a partition. GDH3.0 scales with the size of merging group members for merges and keeps constant for partitions. In the long term, average phases of STR are smallest, followed by EGK, CCEGK, TGDH, GDH3.0.

In Figure 7, when the total partition operations are increasing, average messages of EGK, TGDH, and CCEGK

are increasing slowly; however, average messages of STR remain fairly constants while average messages of GDH3.0 are dropping quickly. From Figure 7, we know rebuilding the tree in EGK is not a bad ideal when the leaving members are very large. Average messages of CCEGK and TGDH are almost equal because they keep constant for merges and scale with the size of leaving group members for partitions. GDH3.0 scales with the size of merging group members for merges and keeps constant for partitions. For the long term, average messages of STR are the smallest, followed by CCEGK and TGDH whose average messages are almost equal, EGK, GDH3.0 with the largest average messages.

In Figures 8 and 9, when the total partition operations are increasing, average sequential exponentiations of GDH3.0 and STR are decreasing rapidly. However, average sequential exponentiations of EGK, TGDH, and CCEGK are increasing steadily. Because in a merge operation, the key tree of CCEGK keeps a little imbalance, the average sequential exponentiations do not increase after we do a lot of partition and merge operations, and EGK and TGDH scale with the height of key tree for merges and partitions. STR and GDH3.0 scale with the group size for merges and partitions. Therefore, when they do partitions, the group size is decreased quickly, average sequential exponentiations are also sharply reduced. In the long term, average sequential exponentiations of EGK are the smallest, followed by CCEGK, TGDH, STR and GDH3.0 with equal and the largest sequential exponentiations.

4.3 Join-leave-mass join-mass leave-merge-partition Results

Scenario	Add	Leave	Mass join	Mass Leave	Merge	Split
1	95	5	0	0	0	0
2	85	5	2	3	2	3
3	75	10	4	3	4	4
4	65	20	4	3	4	4
5	55	25	6	4	5	3
6	45	30	7	5	6	7
7	35	35	8	7	7	8
8	25	40	9	9	8	9
9	15	50	9	9	8	9
10	5	55	10	10	9	11

Table 3: Ten Scenarios in Join-Leave-Mass join-Mass Leave-Merge-Partition

Table 3 shows ten random scenarios used in our simulation to demonstrate the tendency of the performance among the five protocols in the join-leave-mass join-mass leave-merge-partition operation. From Scenario 1 through Scenario 10, the join operation is decreasing and leave, mass join, mass leave, merge, and partition operations are increasing.

Figure 10 shows average phases, and Figures 11 and 12 demonstrate average messages, and Figure 13 and 14 present average sequential exponentiations in the join-leave-mass join-mass leave-merge-partition operation

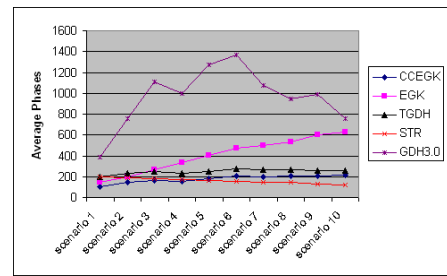


Figure 10: Join-Leave-Mass join-Mass leave-Merge-Partition–Average Phases for five protocols

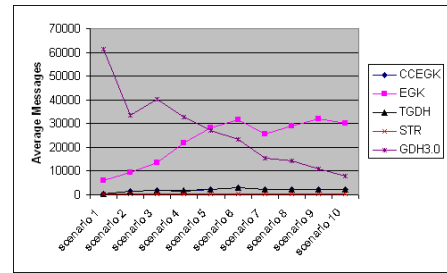


Figure 11: Join-Leave-Mass join-Mass leave-Merge-Partition–Average Messages for five protocols

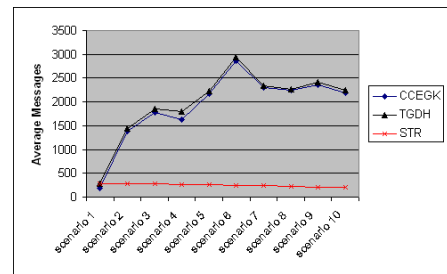


Figure 12: Join-Leave-Mass join-Mass leave-Merge-Partition–Average Messages for the best three protocols

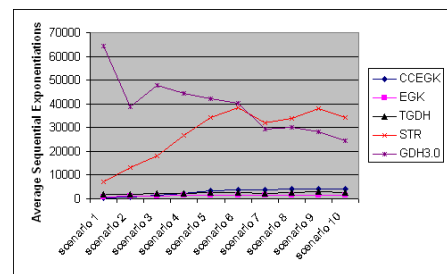


Figure 13: Join-Leave-Mass join-Mass leave-Merge-Partition–Average Sequential Exponentiations for five protocols

when the size of each original group is 600. The reasons of their tendency in the following figures can be summarized by the above explanations of Section 4.1 and Section 4.2

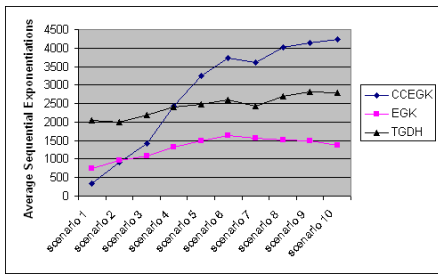


Figure 14: Join-Leave-Mass join-Mass leave-Merge-Partition–Average Sequential Exponentiations for the best three protocols

In Figure 10, when the join operation is decreasing, and leave, mass join, mass leave, merge, partition operations are increasing, average phases in STR are decreasing steadily because in every operation the phases of STR keep constant. However, average phases of CCEGK, TGDH, and EGK are gradually rising while average phases of GDH3.0 are falling steadily. In the long term, average phases of STR are the smallest, followed by CCEGK, TGDH, EGK, GDH3.0.

In Figures 11 and 12, when the join operation is decreasing, and leave, mass join, mass leave, merge, partition operations are increasing, average messages of EGK are increasing quickly; however, average messages of STR remain fairly constant. Average messages of GDH3.0 are falling quickly while average messages of TGDH, and CCEGK are increasing steadily. For the long term, average messages of STR are the smallest, followed by CCEGK, TGDH, EGK, GDH3.0.

In Figure 13 and 14, when the join operation is decreasing, and leave, mass join, mass leave, merge, partition operations are increasing, average sequential exponentiations of GDH3.0 are decreasing rapidly; however, average sequential exponentiations of EGK, TGDH, and CCEGK are increasing steadily while average sequential exponentiations of STR are increasing sharply. In the long term, average sequential exponentiations of EGK are the smallest, followed by TGDH, CCEGK, STR, GDH3.0.

5 Conclusion

In this paper we demonstrated a thorough experimental results showing the average computation and communication costs of five group key management protocols. These experiments are important in understanding the full behavior of these protocols to provide decision maker a better understanding of their performance over time.

From these extensive experiments, the average phases and messages of CCEGK, and STR are most efficient, followed by TGDH, EGK, GDH3.0. Average sequential exponentiations of EGK are the most efficient, followed by TGDH, CCEGK, STR, GDH3.0.

From the above analysis, we understand the relative

strengths and weakness of these protocol. It helps us to decide which protocol will be used in the network for different environment. In the network of lower communication power, CCEGK and STR are selected to provide reliable group communication. In the network requiring lower computation power, EGK, TGDH, and CCEGK are chosen to provide reliable group application. For the network requiring the lower communication and computation power, CCEGK and TGDH are chosen to offer the group communication application.

References

- [1] J. Alves-Foss. An efficient secure authenticated group key exchange algorithm for large and dynamic groups. In *Proc. 23rd National Information Systems Security Conference*, pages 254–256, Oct. 2000.
- [2] Y. Amir, Y. Kim, C. Nita-Rotaru, and G. Tsudik. On the performance of group key agreement protocols. *ACM Transactions on Information and System Security*, 7(3):457–488, Aug. 2004.
- [3] K. Becker and U. Willie. Communication complexity of group key distribution. In *Proc. 5th Conference on Computers and Communication Security*, pages 1–6, 1998.
- [4] M. Burmester and Y. Desmedt. A secure and efficient conference key distribution system. In *Advances in Cryptology - EUROCRYPT'94*, pages 275–286, May 1994.
- [5] M. Burmester and Y. Desmedt. Efficient and secure conference key distribution. In *Security Protocols: International Workshop*, pages 119–129, April 1996.
- [6] I. Ingemarsson, D. Tang, and C. Wong. A conference key distribution system. *IEEE Transactions on Information Theory*, 28(5), Sep. 1982.
- [7] M. Just and S. Vaudenay. Authenticated multi-party key agreement. In *Advances in Cryptology – ASIACRYPT '96*, pages 36–49, 1996.
- [8] Y. Kim, A. Perrig, and G. Tsudik. Group key agreement efficient in communication. *IEEE Transactions on Computers*, 53(7):905–921, Jul. 2004.
- [9] Y. Kim, A. Perrig, and G. Tsudik. Tree-based group key agreement. *ACM Transactions on Information and System Security*, 7(1):60–96, Feb. 2004.
- [10] D. Steer, L. Sawczynski, W. Diffie, and M. Weiner. A secure audio teleconference system. In *Advances in cryptology*, pages 520 – 528, Aug. 1990.
- [11] M. Steiner, G. Tsudik, and M. Waidner. Diffie-hellman key distribution extended to groups. In *Third ACM Conference on Computer and Communications Security*, pages 31–37. ACM Press, Mar. 1996.
- [12] M. Steiner, G. Tsudik, and M. Waidner. Key agreement in dynamic peer groups. Technical report, Information Sciences Institute, Jan. 1999.
- [13] S. Zheng, J. Alves-Foss, and S. Lee. Exploring average performance of group key management algorithms over multiple operations. In *4th IASTED International Conference on Communications, Internet and Information Technology*, number 183-188, Oct. 2005.
- [14] S. Zheng, D. Manz, and J. Alves-Foss. A communication-computation efficient group key algorithm for large and dynamic groups. Technical report, University of Idaho, Sep. 2004.