

EXPLORING AVERAGE PERFORMANCE OF GROUP KEY MANAGEMENT ALGORITHMS 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

Traditional analysis of group key protocol performance is based on the cost of performing a single operation. We extend this analysis to examine the performance behavior of five group key protocols after execution of multiple operation. This paper reports the results of our experiments for 100 operations consist of combinations of join, leave, mass add and mass leave operations. The results of these experiments are consistent with the original single operation experiments, thus validating their utility.

KEY WORDS

group key management, group communications, cryptographic protocols

1 Introduction

The Internet is widely used to support collaborative applications such as conference calls, voice conferencing, distributed computation, white-boards and distributed databases. All of these necessarily need reliable services such as data privacy, integrity, and authentication. Group key management is the most basic security service to provide secure and reliable communication between multiple groups. A number of group key management techniques have been presented to provide computation-communication efficient group key protocol for large and dynamic groups [1, 2, 3, 4, 5, 6, 7]. There are three categories (centralized, distributed, and contributory group key management) that have been put forward to solve group key establishment [8, 9, 10]. In this paper we focus on contributory group key management, in which every group member contributes an equal share to a shared group key. This is appropriate for dynamic peer groups and alleviates the problem of a single point of failure and trust in centralized group key management algorithm. It also avoids long term channel maintenance problems. In this

*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.

paper we analyze the performance of five notable group key protocols based on contributory group key management: GDH3.0 [8], EGK [9], TGDH [11], STR [12], CCEGK [13] 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 the initialization operation in TGDH and STR, concluding with a summary of the theoretical worst-case performance of the five protocols. In Section 3 we discuss the simulation setup, and we present the results of our simulation in Section 4.

2 Related work

In recent years several group key protocols have been introduced, such as GDH3.0 [8], EGK [9], TGDH [11], and STR [12], CCEGK [13] that are based on contributory group key management to improve the cost of communication and computation. The protocols build on the classic two-party Diffie-Hellman (DH) approach for key exchange [14], extending it to groups of parties.

For this to work, several group key management operations, such as initialization, join, mass join, leave, mass leave, merge, partition, and key refresh, are used by group key systems. Communication costs consist of number of rounds, number of unicast messages, number of broadcast messages, and total messages. Computation costs include total sequential exponentiations, total signatures, and total verifications [10, 13]¹.

GDH3.0 is based on a linked list structure, so the number of sequential key exchange operations scales linearly with group size. EGK, TGDH, STR, and CCEGK are based on binary tree structures to provide for an efficient number of exchanges, but they provide different algorithms for the group key and group management protocols. EGK and TGDH focus more on reducing the cost of computation than that of communication. STR focuses on improving the

¹The DH two-party key exchange requires exponentiation to provide cryptographic strength. This exponentiation is typically over very large integers (1024 bits) using modular arithmetic, and is thus time consuming. The group key algorithms often use results of one operation as input to the other, hence the use of *sequential exponentiations* as a measure of performance.

cost of communication more than the cost of computation. CCEGK attempts to balance communication and computation costs.

2.1 Theoretical Comparison

Table 1 summarizes the theoretical worst case performance costs of the algorithms we are analyzing. Full performance metrics typically include the cost of communication (number of rounds, number of messages, number of unicast messages, number of broadcast messages) and the cost of computation (number of sequential exponentiations, signatures, and verifications) [7]. However, due to the fact that not all operations are fully documented in the literature, we only present the costs for five operations (initialization, join, mass join, leave, mass leave). The performance of the initialization operation for TGDH and STR is based on our best guess of their implementation of this operation, since it is not fully documented.

We use n to denote the sizes of the current group, and K to denote the number of mass join members or mass leave members. We use h to indicate the height of the current key tree, $h' = \lceil \log_2 K \rceil$ to denote merging key tree, and define $\rho = \min(h' + 1, h)$, and $\mu = \min(2K, n - K)$.

Table 1 lists the worst case costs of the evaluation metrics among these five protocols. It is important to see how these protocols operate when multiple operations are executed. In Section 3 we therefore compare the average cost of multiple instances of these operations, through simulation, to get a better feel for the algorithms.

3 Simulation Setup

Amir et al. compare the performance of every single operation among GDH3.0, BD, CKD, TGDH, and STR for different group sizes [10]. Zheng et al. compare average communication and computation costs of every operation among EGK, STR, TGDH and CCEGK for different group sizes [13]. The drawback of these 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 [10]. However, for the leave operation, TGDH outperforms other protocols. From these results, it is hard to determine whether TGDH is more efficient than STR, because we do not know what happens when we execute join and leave operations together.

In our work, we consider the average cost of evaluation metrics in combined operations among GDH3.0, EGK, STR, TGDH, and CCEGK. In the simulations reported in this paper, there are two categories of group operations: **join-leave** and **mass join-mass leave**.

Join-leave, a combination of the two single member operations join or leave, is executed every time. In our simulations, we assume that the occurrence of join and leave

operations are independent and uniformly distributed.

Mass join-mass leave, an operation that is one of two operations: mass join or mass leave, is run every time. In our tests, we assume that the occurrence of mass join and mass leave operations are uniformly distributed and independent.

3.1 Test Scenarios

The costs of communication and computation in each protocol rely on a number of factors. We design our simulations to 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. EGK's communication and computation costs do not depend on the position(s) of the joining or leaving member(s) because in join and mass join operations, it adds the new member(s) to the root of the original key tree, and in the initialization, leave and mass leave operations, a new balanced binary tree is reconstructed. The communication and computation costs of TGDH and CCEGK depend on: the position(s) of the leaving or joining member(s), tree height, and the balancedness of the tree. STR's costs depend on the position(s) of leaving member(s). Based on this discussion, we restricted our simulations as follows:

- for GDH 3.0, 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 initialization operations) to create a tree structure that could possibly exist at some point in time.
- for STR, we obey the specified rules to create the binary tree using an initialization operation.

We initially conducted our experiments with initial group sizes 200, 600 and 1000. We found that the relative performance of the protocols was the same in these groups and therefore only present the results for groups of 600 in this paper. For each experiment of two operations we ran 100 operations. Each operation was chosen to be either a joining or leaving operation based on a probabilistic paring $\{(0\%, 100\%), (10\%, 90\%), \dots\}$. The charts are graphed based on the probability of the leaving operation. To summarize, our experimental configuration was:

- total operations = 100 (probabilistically distributed)
- initial group size = 600
- results presented are averages over 10 runs
- for leave experiments the leaving member is chosen randomly over all members

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	$K+1$	0	$K+1$	K	1	$K+1$
	Leave	1	1	0	1	$3h-3$	1	1
	Mass Leave	ρ	μ	0	μ	$3h-3$	ρ	μ
EGK	Initialize	h	$2n-2$	0	$2n-2$	$2h-2$	h	$2n-2$
	Join	1	2	0	2	1	1	2
	Mass Join	$h'+1$	$2K$	0	$2K$	$2h'$	$h'+1$	$2K$
	Leave	h	$2(n-1)$	0	$2(n-1)$	$2h$	h	$2(n-1)$
	Mass Leave	h	$2(n-K)$	0	$2(n-K)$	$2h$	h	$2(n-K)$
TGDH	Initialize	h	$2n-2$	0	$2n-2$	$2h-2$	h	$2n-2$
	Join	2	3	0	3	$3h-3$	2	3
	Mass Join	$h'+1$	$2K$	0	$2K$	$3h-3$	$h'+1$	$2K$
	Leave	1	1	0	1	$3h-3$	1	1
	Mass Leave	ρ	μ	0	μ	$3h-3$	ρ	μ
STR	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	$K+2$	0	$K+2$	$3K+1$	2	$K+2$
	Leave	1	1	0	1	$\frac{3n}{2}+2$	1	1
	Mass Leave	1	1	0	1	$\frac{3n}{2}+2$	1	1
GDH3.0	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	$K+3$	$n+2K+1$	0	$n+2K+1$	$n+2K+1$	$K+3$	$n+2K+1$
	Leave	1	1	0	1	$n-1$	1	1
	Mass Leave	1	K	0	K	$n-K$	1	1

Table 1: Performance Metrics for Five Group Key Agreement Protocols

- for mass join, we randomly chose from 2 to 10 members to join
- for mass leave, we randomly chose from 2 to 10 members to leave

In each of the group operations, we compute the averages for phases, messages, sequential exponentiations, signatures, and verifications. Since average verifications are the same as average messages, and average signatures are the same as 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 Results

Figure 1 shows the average number phases. We use both Figure 2 and 3 to depict the average number of messages. Due to the large disparity in cost performance, Figure 3 shows only the three best performing protocols. Figures 4 and 5 present the average sequential exponentiations, with the best three isolated in Figure 5.

In Figure 1, when the total leave operations are increasing, average phases in TGDH, STR, and GDH3.0 are decreasing slightly because a leave operation of TGDH, STR, or GDH3.0 only needs one phase, a join operation

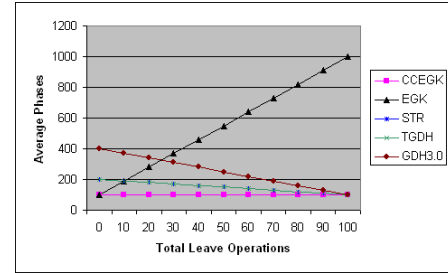


Figure 1: Join-Leave–Average Phases

of TGDH, or STR requires two phases, and a join operation of GDH3.0 needs four phases. However, average phases in EGK are greatly increasing while average phases in CCEGK remain constant. This is because EGK was designed to be efficient for joins and used a built-in balancing mechanism for leaves, by rebuilding the tree after every leave. The experimental result here clearly demonstrates that this rebalancing was not a good idea if a system experiences large numbers of leaving members.

Since the probabilities of occurrences of leave and join operation are equal for the long term we know eventually the total leave and join operations will be equal. At this equality point, the phases of CCEGK in the join-leave operation are the smallest, followed by STR and TGDH,

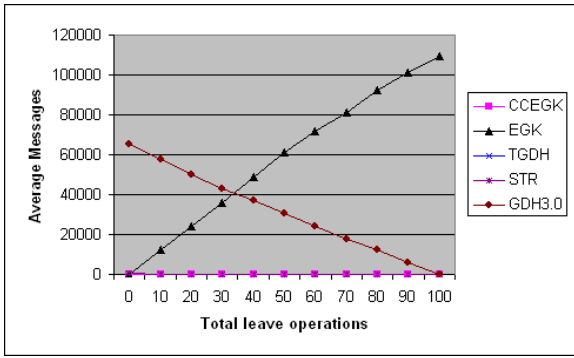


Figure 2: Join-Leave-Average Messages

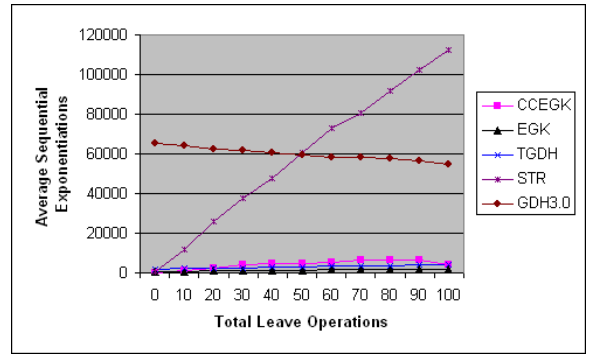


Figure 4: Join-Leave-Average Sequential Exponentiations

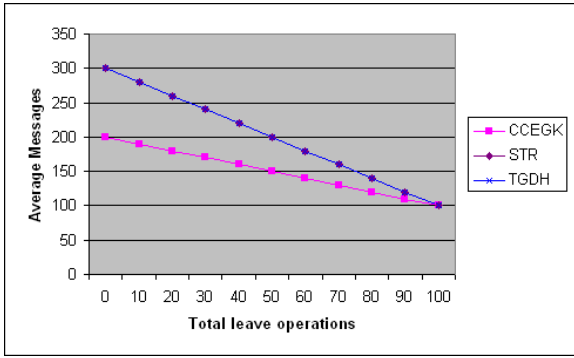


Figure 3: Join-Leave-Average Messages for Best Protocols

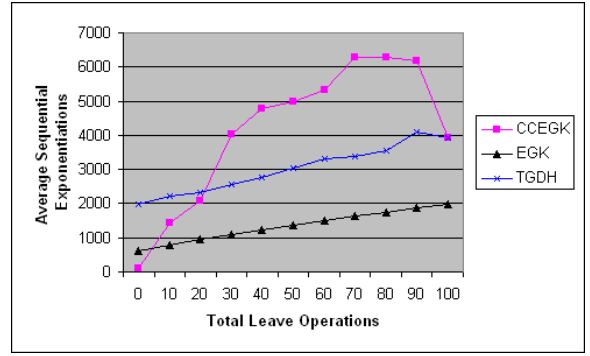


Figure 5: Join-Leave-Average Sequential Exponentiations for Best Protocols

and then GDH3.0, and EGK.

In Figures 2 and 3, when the total leave operations are increasing, messages of EGK are climbing dramatically because during the leave operation the tree is reconstructed and needs $2n' - 2$ messages (where n' is the size of the updated group), messages of GDH3.0 are decreasing sharply because a leave operation needs one message, and a join operation needs $n + 3$ messages. Messages of TGDH, STR, and CCEGK are decreasing slowly because in a join operation, TGDH and STR need three messages while CCEGK needs two messages, and in a leave operation, TGDH, STR, and CCEGK need one message. From these two figures, in the long term, average messages of CCEGK are the smallest, followed by TGDH and STR and then the others.

In Figure 4 and 5, when the total leave operations increase, sequential exponentiations in GDH3.0 go down slowly because in the join operation, GDH3.0 needs $n + 3$ sequential exponentiations; however, sequential exponentiations in STR jump quickly because in an join operation, STR only needs four sequential exponentiations and in a leave operation it needs $n + 3$. Sequential exponentiations of EGK, and TGDH rise steadily because in a join operation, EGK needs one sequential exponentiation and TGDH needs at most $3h - 3$. In a leave operation, EGK reconstructs the tree and requires $2h - 2$ sequential exponentiations and TGDH still needs at most $3h - 3$. Sequen-

tial exponentiations of CCEGK increases steadily until the number of leave operations exceeds the number of joins, because when CCEGK does more join operations, the tree become more imbalanced, which will increase the total sequential exponentiations. In this paper we do not consider the rebalance scheme of CCEGK in our simulation. For the long term, average sequential exponentiations of EGK are the smallest, followed by TGDH, CCEGK, STR and GDH3.0.

4.2 Mass Join - Mass Leave Results

Figure 6 represents average phases, Figures 7 and 8 demonstrates average messages, and Figures 9 and 10 illustrate average sequential exponentiations in the **mass join-mass leave** operation when the group member size is 600 and every time the joining or leaving members are between 2 and 10.

In Figure 6, when the total mass leave operations are increasing, average phases in STR and GDH3.0 are decreasing steadily because during a mass join operation, STR needs two phases and GDH3.0 requires $K - 3$ (K is the number of joining members), and during a mass leave operation, STR and GDH3.0 only need one phases. However, phases in EGK and CCEGK are gradually increasing because during a mass join operation, EGK needs

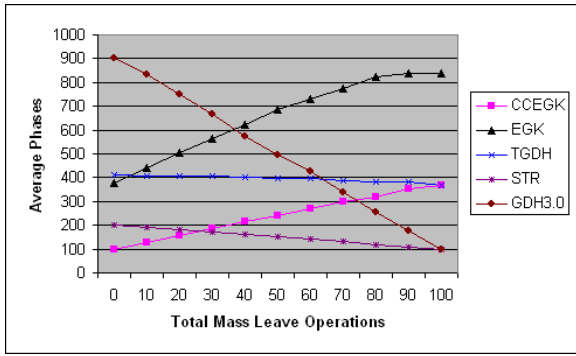


Figure 6: Mass Join-Mass Leave–Average Phases

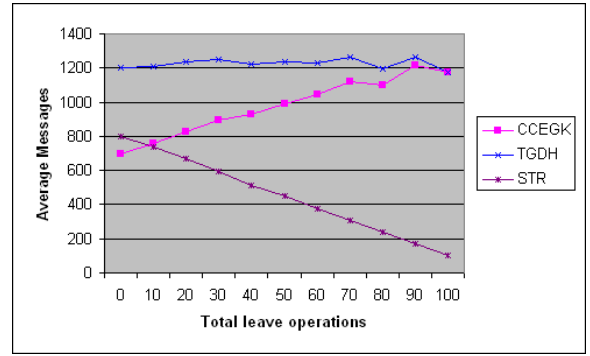


Figure 8: Mass Join-Mass Leave–Average Messages for Best Protocols

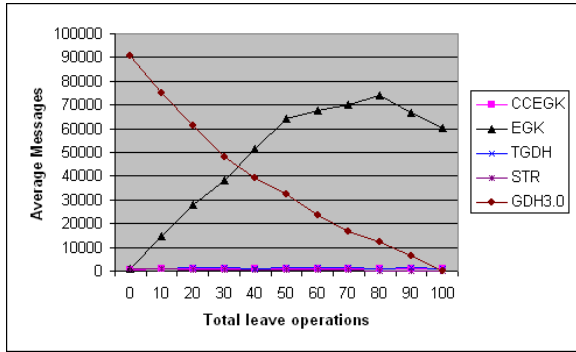


Figure 7: Mass Join-Mass Leave–Average Messages

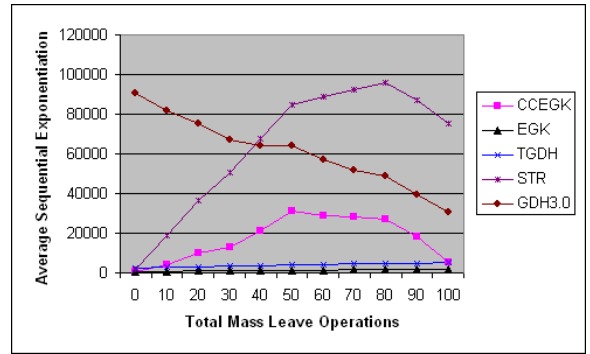


Figure 9: Mass Join-Mass Leave–Average Sequential Exponentiations

$\lceil \log_2 K \rceil + 1$ phases, and CCEGK needs one phase and during a mass leave operation EGK reconstructs the tree and needs $2h - 2$, and CCEGK needs at most $\min(\lceil \log_2 K \rceil + 1, h)$ phases. The phases of TGDH are fairly stable because in a mass join operation it needs $\lceil \log_2 K \rceil + 1$, and during a mass leave operation it requires at most $\min(\lceil \log_2 K \rceil + 1, h)$. In the long term, the average phases of STR are the smallest, followed by CCEGK, TGDH, GDH3.0 and EGK.

In Figures 7 and 8, when the total mass leave operations are increasing, average messages of EGK are increasing dramatically because during a mass join operation it needs $2K$ messages, and during a mass leave operation it requires $2n' - 2$ messages (n' is new size of the updated group). However, the average messages of GDH3.0 are decreasing rapidly because in a mass join operation it needs $n + 2K + 1$ messages, and during a mass leave operation it only requires K messages. Average messages of STR decrease quickly because during a mass join operation it needs $K + 2$ messages, and during a mass leave operation it only requires one message. Average messages of CCEGK rise steadily because in a mass join operation it needs $K + 1$ messages, and during a mass leave operation it requires at most $\min(2K, n - K)$. Average messages of TGDH remain steady because in a mass add operation it needs $2K$, and in a mass leave operation it requires $\min(2K, n - K)$. For the long term, average messages of STR are the smallest,

followed by CCEGK, TGDH, GDH3.0 and EGK.

In Figures 9 and 10, when the total mass leave operations are increasing, average sequential exponentiations in GDH3.0 are decreasing rapidly because during a mass join operation, it needs $n + 2K + 1$, and during a mass leave operation, it requires $n - K$. However, average sequential exponentiations in TGDH and EGK are increasing steadily because in a mass join operation TGDH needs at most $3h - 3$ and EGK needs $2 \lceil \log_2 K \rceil$, and during a mass leave operation, TGDH still needs at most $3h - 3$, and EGK needs $2h - 2$. Average sequential exponentiations of STR and CCEGK are increasing sharply in the first half of total operations. After that, average sequential exponentiations of STR are still increasing gradually and then decreasing steadily while average sequential exponentiations of CCEGK are dropping rapidly. The reason is that in a mass join operation, STR needs $3K + 1$ and CCEGK needs K , and in a mass leave operation, STR requires $\frac{3n}{2} + 2$ and CCEGK requires at most $3h - 3$. In the long term, average sequential exponentiations of EGK are the smallest, followed by TGDH, CCEGK, GDH3.0 and STR.

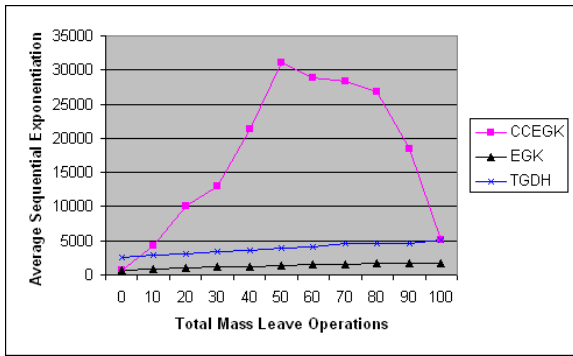


Figure 10: Mass Join-Mass Leave–Average Sequential Exponentiations for Best Protocols

5 Conclusion and Future Work

In this paper we presented experimental results showing the average computation and communication costs of five group key management protocols. These experiments are an essential first step in understanding the full behavior of these protocols to provide decision makers a better understanding of their performance over time.

From the above two group operation simulations (join-leave and mass join-mass leave), we know the average phases and messages of CCEGK and STR are the most efficient, followed by TGDH, GDH3.0, EGK. The average sequential exponentiations of EGK is the most efficient, followed by TGDH, CCEGK, GDH3.0 and STR.

In the future, we will implement partition operations in TGDH and STR. After that, we will simulate three extra group operations: join-leave-mass join-mass leave, merge-partition, join-leave-mass join-mass leave-merge-partition to further compare the performance of these five protocols. We hope to utilize the results of these experiments to better understand the relative strengths and weaknesses of these protocols.

References

- [1] I. Ingemarsson, D. Tang, and C. Wong. A conference key distribution system. *IEEE Transactions on Information Theory*, 28(5), Sep. 1982.
- [2] D. Steer, L. Sawczynski, W. Diffie, and M. Weiner. A secure audio teleconference system. In *on Advances in cryptology*, pages 520 – 528, Aug. 1990.
- [3] M. Burmester and Y. Desmedt. A secure and efficient conference key distribution system. In *Advances in Cryptology - EUROCRYPT'94*, pages 275–286, May 1994.
- [4] M. Burmester and Y. Desmedt. Efficient and secure conference key distribution. In *Security Protocols: International Workshop*, pages 119–129, April 1996.
- [5] M. Just and S. Vaudenay. Authenticated multi-party key agreement. In *Advances in Cryptology – ASIACRYPT '96*, pages 36–49, 1996.
- [6] 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.

- [7] K. Becker and U. Willie. Communication complexity of group key distribution. In *Proc. 5th Conference on Computers and Communication Security*, pages 1–6, 1998.
- [8] M. Steiner, G. Tsudik, and M. Waidner. Key agreement in dynamic peer groups. Technical report, Information Sciences Institute, Jan. 1999.
- [9] 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.
- [10] 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.
- [11] 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.
- [12] Y. Kim, A. Perrig, and G. Tsudik. Group key agreement efficient in communication. *IEEE Transactions on Computers*, 53(7):905–921, Jul. 2004.
- [13] 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.
- [14] W. Diffie and M. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644–652, Nov. 1976.