중소기업융합학회 논문지 제4권 제4호 pp. 19-23, 2014 ISSN 2234-4438

Efficient and Secure Group Key Generation Protocol for Small and Medium Business

Jung Hyun Soo[1*]
[1*]Information Security, Songsil University

Abstract Group communication is becoming increasingly popular in Internet applications such as videoconferences, online chatting programs, games, and gambling. For secure communications, the integrity of messages, member authentication, and confidentiality must be provided among group members. To maintain message integrity, all group members use the Group Key (GK) for encrypting and decrypting messages while providing enough security to protect against passive attacks. Tree-based Group Diffie-Hellman (TGDH) is an efficient group key agreement protocol to generate the GK. TGDH assumes all members have an equal computing power. One of the characteristics of distributed computing and grid environments is heterogeneity; the member can be at a workstation, a laptop or even a mobile computer. Member reordering in the TDGH protocol could potentially lead to an improved protocol; such reordering should capture the heterogeneity of the network as well as latency. This research investigates dynamic reordering mechanisms to consider not only the overhead involved but also the scalability of the proposed protocol.

Key Words : Protocol design, Group key management, Network security, Secure group communication.

1. Introduction

Group communications are created all over the network in the form of videoconferences, on-line chatting programs, games, and gambling. Security plays an important role in these instances of group communication. According to [3, 4], member authentication processes and key distribution take place at the beginning of a group communication. The group size tends to be less than 100 [8]. However, the Group Key (GK) computation takes a relatively long time to complete. 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 [3]. The group key agreement protocol focuses on the GK computation, which consists of evaluating a function of modular exponentiations. In order to calculate the GK using modular exponentiations, the adaptation of key trees is needed to reduce the computational overhead. Modular exponentiation is the computationally most expensive operation in TGDH [2]. The number of exponentiations for membership events depends on the number of group members. The algorithm efficiency of TGDH is $O(\log_2 n)$, where n is the current number of members, so it is efficient as long as the key tree is perfectly balanced. However, maintaining a perfect key tree balance results in a significant overhead. Maintaining a perfectly balanced tree after a membership change is one problem; another is that TGDH assumes an underlying homogeneous network. However, in distributed computing environments, members can be at a workstation, a laptop, or even a mobile computer.

Received 2014-11-01 Revised 2014-11-20 Accepted 2014-11-27
*Corresponding author: Jung Hyun Soo (hsj6552@hanmail.net)

For example, if there is a mobile computer member in the group communication and his position is assigned to the last position in the key computational sequence in the group key computation, then he/she must calculate the cardinal value $K = g^{K1K2...Kn} \bmod p$. The last positioned member has to compute the most complex value, so if a mobile member is assigned to the last position in the key computation, then all members must wait longer for obtaining the GK. It is clear that a member sequence must be reordered taking into account that the network is composed mainly of heterogeneous components and therefore, each has different computing power. A reordering scheme to optimize the GK computation is proposed in this research. A secure and efficient key management is a critical issue in group communication [7]. If the system's performance is low, then the system's usability will be low. Therefore, the focus on increasing the efficiency of the group key computation is aimed at maximizing system's usability.

2. Related Work

Group communication arises in many different settings, from low-level network multicasting to conferencing, and other groupware applications. Regardless of the environment, security services are needed to provide communication privacy and integrity. These services are not possible without a secure and efficient key distribution, authentication, and other mechanisms. In a secure communication, group members need a common group key to protect their messages exchanged as well as group key management for the computation and distribution of the GK. Unless the communication channel is secure, delivery of messages over the network to the right destination cannot be guaranteed. Group key management is a building block to provide such assurance. There are two types of schemes in group key management, *group key distribution* and *group key agreement* [9]. The group key distribution is assigned to one member in the group who then becomes the key distribution center. He/she computes the GK and distributes it to each member in the group. The group key agreement is suitable for peer-to-peer group communication [9]. In these groups the group key agreement protocol ensures that each member has an equal opportunity for generating the GK. One member takes the role of the Group Controller (GC), collects all the members' blind keys (public keys), broadcasts the group key computation tree structure to all members, and controls the overall group key computational processes [12, 13].

3. Tree-based Diffie-Hellman Group Key Computation

The Group Diffie-Hellman (GDH) key agreement protocol [10, 11] is an extension to the Diffie-Hellman (DH) key exchange protocol [3]. The GK computation is an important component of group key management in securing group communication; several efforts to enhance the group key computational process have been reported [13,14,15] 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 [1]. Because of the complexity of the GK computation, the group key management adopts 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 calculation [6]. One such group key computational protocol is the Tree-based Group Diffie-Hellman TGDH [5].

An example of the key tree-based GK computational process follows. In the binary key tree for generating a group key in Fig. 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 $BK_{<l, v>}=f(K_{<l, v>}) = g^{K<l, v>} \bmod p$, where g and p are large 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 Fig. 1. The final group key K<0,0> in Fig. 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$BK_{<3,\ 7>}$[2]. Therefore, the final group key can be computed as Eq.(1):

$$K_{<0,0>} = g^{(g^{g^{K_{<3,0>}K_{<3,1>}g^{K<2,1>}}})(g^{g^{K_{<2,2>}g^{K<3,6>K_{<3,7>}}}})} \bmod p \qquad (1)$$

The TGDH has two major disadvantages. First, maintaining a balanced group key computational tree causes overhead. The group key computational tree must be balanced at any given time so that the efficiency of group key computation would be *O(log n).* Otherwise, the performance of group key computational key would be worse than *O(log n).* The second disadvantage is derived from no regard for member's diversity in that if a slow member such as a mobile computer joins the group key computational processes, then the wait time would be long.

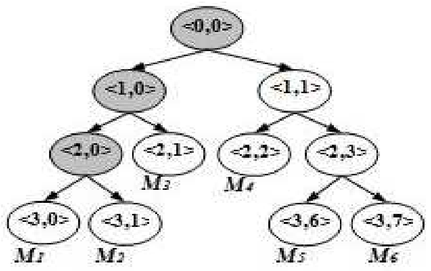


Fig. 1. A Binary Tree for Group Key

4. Performance Evaluations

Big O notation is useful when analyzing algorithms for efficiency. For example, the time (or the number of steps) it takes to complete a problem of size n might be found to be $T(n) = 4n^2 - 2n + 2$. As n grows large, the n^2 term will come to dominate, so that all other terms can be neglected [14]. The efficiency of T(n) is O(n^2), so we can compare algorithm efficiency with Big O notation. Complexity analysis requires however, that the problem size n must be significantly large. As mentioned earlier, the group size is less than 100, so it is not suitable to analyze group key computational protocols. Therefore, direct measurements is the only way to estimate elapsed times for comparing the algorithm efficiency.

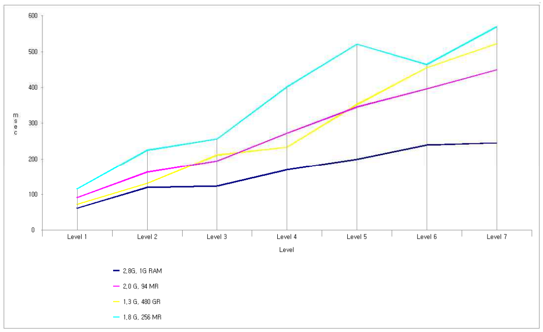
In this section, group computational processes were tested at four different Intel Pentium IV machines. Each machine's elapsed times were measured 16 times for the group key computation at each level in Table 1. Their averages are found in Table 1. We used 1,024-bit integer g (exponentiation base), p (divider), and K (secret key) for all measurements. These values are known to be secure in the current technology [15].

Fig. 2 is based on Table 1. The values in x-axis mean the level of group key computation. The values in y-axis are elapsed times (msec). TGDH was compared with Enhanced Group Key Computation Protocol (EGKCP). Fig. 3 shows the differences between TGDH and EGKCP. TGDH does not consider member's computing power. Thus, TGDH must wait until the slowest member has completed computation of the group key.

However, in EGKCP, only fast members are allowed to compute the group key, so it always takes the least time to compute the group key. Therefore, the overall performance of EGKCP is on average 2.9 times faster than in TGDH.

Table 1. Computational time for group key (msec)

Level	2.8G, 1G R	2.0 G, 94 MR	1.3G, 480 MR	1.8G, 256 MR
1	61.0	91.0	72.0	115.0
2	119.0	162.0	131.0	224.0
3	123.0	194.0	211.0	256.0
4	169.0	272.0	233.0	401.0
5	199.0	345.0	352.0	521.0
6	239.0	396.0	455.0	464.0
7	245.0	449.0	522.0	569.0



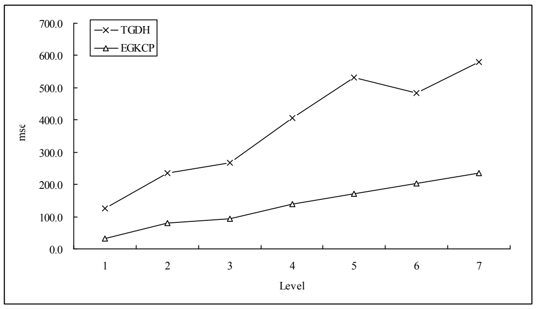


Fig. 3. Group Key Computation for Each Machine

5. Conclusions

Group key computation protocols must consider a variety of members; otherwise, the system usability will be degraded. Currently mobile computers are becoming more popular. Network clusters are communicating with conventional servers. The enhanced group key computation protocol is proposed because conventional group key agreements do not consider the network heterogeneity in terms of member's computational power. The enhanced protocol proposed also takes into account the latency of the network as it affects message delays during the group interaction. Each time membership changes, the members who join in the group key computational processes must be fast members to avoid unexpected delays in obtaining the group key. The BKQ structure proposed is being used to order messages containing blind (public keys) in the order of arrival. The first members in the queue are selected to join in the group key computational processes. Currently the feasibility of the Enhanced Group Key Computation Protocol (EGKCP) approach is being investigated in terms of improved GK efficiency. In addition, the feasibility of the proposed protocol is being investigated in Grid environments [16].

REFERENCES

[1] Y. Kim, A. Perrig, and G. Tsudik, "Tree-based group key agreement," ACM Transaction on Information and System Security, 2004.

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

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

[4] Y. Kim, A. Perrig, and G. Tsudik, “Simple and fault-tolerant key agreement for dynamic collaborative groups,” In S. Jajodia, editor, 7th ACM Conference on Computer and Communications Security, ACM Press, Athens, Greece, pp. 235 - 244, Nov. 2000.

[5] D. Wallner, E. Harder, and R. Agee, “Key management for multicast: Issues and architecture,” Internet-Draft draft-wallner-keyarch-00.txt, Jun. 1997.

[6] Y. Amir, Y. Kim, C. Nita-Rotaru, J. Schultz, and J. Stanton, “Secure group communication using robust contributory key agreement,” IEEE Transaction on parallel and distributed systems, vol. 15, no.4, Apr. 2004.

[7] M. Steiner, G. Tsudik and M. Waidner, “Key agreement in dynamic peer groups,” IEEE Transactions on Parallel and Distributed Systems, vol. 11, no. 8, pp.769-780, Aug. 2000.

[8] Y. Kim, “Group key agreement: theory and practice,” Ph.D. thesis, May. 2002.

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

[10] M. Steiner, G. Tsudik, and M. Waidner, “Cliques: A new approach to group key agreement,” IEEE ICDCS'98 , May. 1998.

[11] A. Fekete, N. Lynch, and A. Shvartsman, “Specifying and using a partionable group communication service,” ACM Transactions on Computer Systems, vol. 19, no. 2, May. 2001.

[12] Y. Amir,; Y. Kim, and C. Nita-Rotaru, "On the performance of group key agreement protocols," ACM transactions on information and system security, vol. 7, no. 3, pp. 457, 2004.

[13] Donald Knuth. "The Art of Computer Programming," vol.1: Fundamental Algorithms, Third Edition. Addison-Wesley, pp. 107–123, 1997.

[14] A. K. Lenstra and E. R. Verheul. Selecting cryptographic key sizes. http://www.cryptosavvy.com/, Nov. 1999. Shorter version of the report appeared in the proceedings of the Public Key Cryptography Conference (PKC2000) and in the Autumn '99 PricewaterhouseCoopers CCE newsletter. To appear in Journal of Cryptology.

[15] Lan Foster, The Grid: Blueprint for a New Computing Infrastructure, Second Edition, Elsevier, pp.47-53, 2004.

저 자 소 개

정 현 수(Jung Hyun Soo) [정회원]

▪ 1982년 2월 : 숭실대학교 전자계산학과 학사
▪ 1991년 2월 : 숭실대학교 컴퓨터학과 석사
▪ 1995년 2월 :숭실대학교 컴퓨터학과 박사

▪ 1982년 2월 ~ 2005년 11월 : ETRI 책임연구원
▪ 2006년 2월 ~ 2011년 3월 : TANC CTO
▪ 2009년 2월 ~ 2012년 2월 : 한남대학교 경영정보학과 겸임교수
▪ 2012년 4월 ~ 현재 : 숭실대학교 정보보안학과 교수

<관심분야> : 정보보호 정책, 컴퓨터 비젼