

# Enhanced Group Key Generation Protocol

Sunghyuck Hong and Noe Lopez-Benitez

Department of Computer Science  
Texas Tech University  
Lubbock, Texas, USA

**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.

**Keywords:** 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], 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. 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  $K = g^{k_1 k_2 \dots k_n} \text{ mod } p$ . 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.

### 3. Tree-Based Diffie-Hellman Group Key Computation

The Group Diffie-Hellman (GDH) key agreement protocol [10] 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] 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  $\langle l, v \rangle$  represents a  $v$ -th node at level  $l$  in the tree and node  $\langle l, v \rangle$ 's secret (private) key  $K_{\langle l, v \rangle}$  and a blind (public) key  $BK_{\langle l, v \rangle} = f(K_{\langle l, v \rangle}) = g^{K_{\langle l, v \rangle}} \text{ mod } p$ , 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  $\langle 0,0 \rangle$ ,  $\langle 1,0 \rangle$ , and  $\langle 2,0 \rangle$ ) in Fig. 1. The final group key  $K_{\langle 0,0 \rangle}$  in Fig. 1 is computed with the key paths using blind keys  $BK_{\langle 3,0 \rangle}$ ,  $BK_{\langle 3,1 \rangle}$ ,  $BK_{\langle 2,1 \rangle}$ ,  $BK_{\langle 2,2 \rangle}$ ,  $BK_{\langle 3,6 \rangle}$ , and  $BK_{\langle 3,7 \rangle}$  [2]. Therefore, the final group key can be computed as Eq.(1):

$$K_{\langle 0,0 \rangle} = g^{(g^{g^{K_{\langle 3,0 \rangle} K_{\langle 3,1 \rangle} g^{K_{\langle 2,1 \rangle}})}} (g^{g^{K_{\langle 2,2 \rangle} g^{K_{\langle 3,6 \rangle} K_{\langle 3,7 \rangle}})}} \text{ mod } p \quad (1)$$

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.

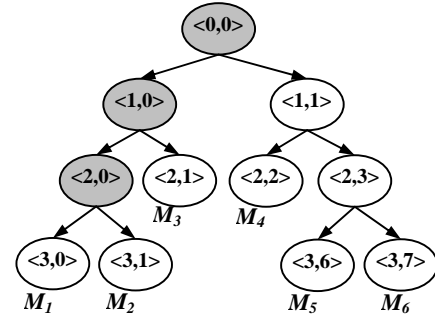


Figure 1. A Binary Tree for Generating Group Key

### 4. Enhanced Group Key Computation Protocol (EGKCP)

The idea of the proposed protocol relies on the premise that the members who participate in distributed computing do not have equal computing power. The proposed key computational protocol requests only members with fast machines join in the group key computational processes to increase efficiency. The group key must be regenerated for secure communication whenever group membership changes. The details of the proposed procedure are described below.

Suppose that the group size is  $n$  and group members are  $M_1, M_2, M_3, \dots, M_{n-1}, M_n$  for  $n < 100$ . If a new member,  $M_{n+1}$  joins the group, then  $M_{n+1}$  becomes the group controller (GC) responsible for controlling the group key computational processes, so any member can be the GC and the GC can store other's blind keys into his / her own Blind Key Queue (BKQ) in Fig. 2. Each member is required to have a Queue structure.

The GC requests all members to generate a blind key ( $g^{K_i} \text{ mod } p$ ,  $i \in [1, n]$ ,  $g$  is an exponentiation base,  $p$  is a prime number, and  $K_i$  is  $i^{\text{th}}$  member's private key) by sending a request message. The group controller (GC) receives all blind keys and stores them into his/her own BKQ in the order of their arrival to determine fast members joining in the next level of the group key computational process. The fastest member's blind key is always stored in the front of the BKQ. On the other hand, the slow members' blind keys go in the rear of the BKQ.

In general, two factors must be considered to determine faster members. One is to measure member's computing

Table 1. Key Computational Processes

| Level | Group key computation   |
|-------|---|
| 1     | $g^{K_1} \bmod p$ (g and p are 1,024-bit integer numbers)   |
| 2     | $g^{K_1 K_2} \bmod p$   |
| 3     | $g^{g^{K_1 K_2} g^{K_3 K_4}} \bmod p$   |
| 4     | $g^{g^{g^{K_1 K_2} g^{K_3 K_4} g^{K_5 K_6} g^{K_7 K_8}}} \bmod p$   |
| 5     | $g^{g^{g^{g^{K_1 K_2} g^{K_3 K_4} g^{K_5 K_6} g^{K_7 K_8} g^{K_9 K_{10}} g^{K_{11} K_{12}} g^{K_{13} K_{14}} g^{K_{15} K_{16}}}}} \bmod p$                |
| 6     | $g^{g^{g^{g^{g^{K_1 K_2} g^{K_3 K_4} g^{K_5 K_6} g^{K_7 K_8} \dots g^{K_{25} K_{26}} g^{K_{27} K_{28}} g^{K_{29} K_{30}} g^{K_{31} K_{32}}}}} \bmod p$    |
| 7     | $g^{g^{g^{g^{g^{g^{K_1 K_2} g^{K_3 K_4} g^{K_5 K_6} g^{K_7 K_8} \dots g^{K_{57} K_{58}} g^{K_{59} K_{60}} g^{K_{61} K_{62}} g^{K_{63} K_{64}}}}} \bmod p$ |

power and the other is to measure communication latency. If the time taken for communicational messages to traverse the network is long, then the member would be regarded as a low performance member no matter how fast their computing power. Therefore, measuring member's performance must include both the computing power and communication latency simultaneously. Therefore, using the BKQ is a simple way to measure both elapsed times for computing the group key and communication latency.

The number of levels in the group key computational processes 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, each member exchanges their blind key with members located on the opposite side of the BKQ in Fig. 2. Following each level of group key computation, the GC collects all computed partial group keys and stores them into the BKQ at every level of the group key computational process. The blind keys are stored in the BKQ in the order of their arrival. Therefore, Fig. 2 describes that the fastest member's key goes to the  $A_1$  spot, the second fastest member's key goes to 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,  $A_n$ . Thus,  $A_1$  spot's blind key ( $g^{K_2} \bmod p$ ) will be computed with  $A_n$  spot's blind key ( $g^{K_{n-1}} \bmod p$ ),  $A_2$  spot's blind key ( $g^{K_3} \bmod p$ ) and so on.

After having completed all levels, the final group key can be computed as:

$$g^{\dots g^{g^{K_3 K_n} g^{K_2 K_{n-1}} \dots g^{K_{(n/2)} K_{n-(n/2)}}} \bmod p \quad (2)$$

In this case, only fast members (the shaded area in Fig. 2) are allowed to join in the next level in the group computational processes. Therefore, the proposed protocol can prevent unnecessary delays and improve efficiency.

## 5. Experimental Results and Comparison

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.

Table 1 shows the levels of group key computational processes, and the higher levels take more computational times to complete the calculation. If a group size is 64, the actual group key computational processes are followed in Table 1.

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 2. We used 1,024-bit integer  $g$  (exponentiation base),  $p$  (divisor), and  $K$  (secret key) for all measurements. These values are known to be secure in the current technology [15].

Fig. 3 is based on Table 2. 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. 4 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.

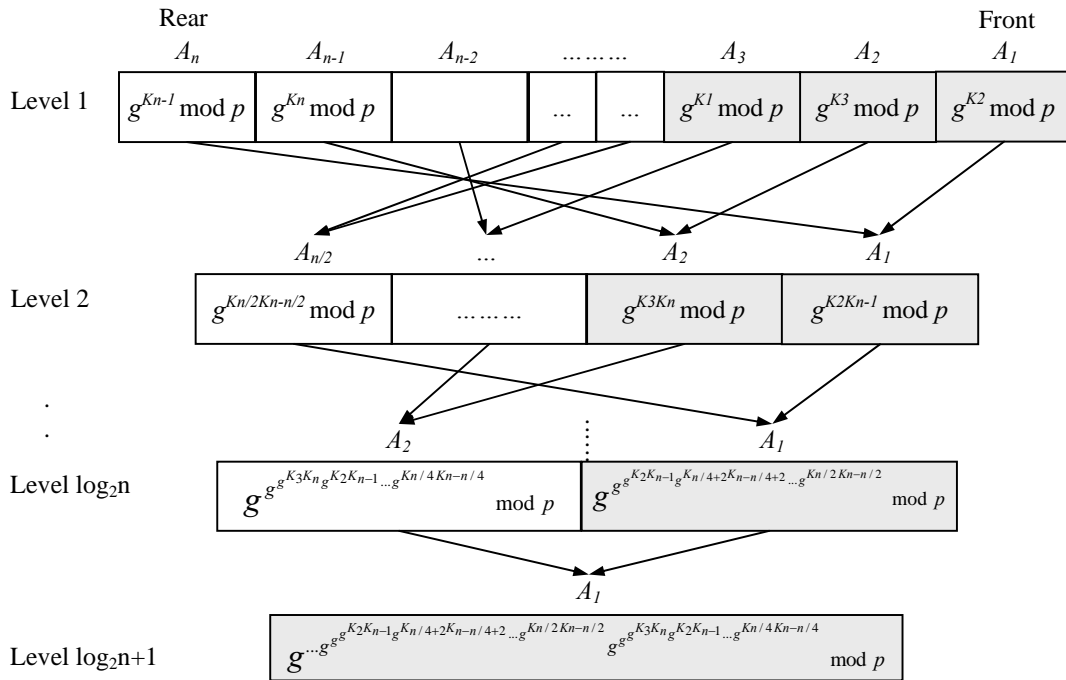


Figure 2. The Blind Key Queue (BKQ) in Group Controller

Table 2. Average elapsed time for group key computation (msec)

| Level | 2.8G,<br>1G R | 2.0 G,<br>94 MR | 1.3G,<br>480 MR | 1.8G,<br>256 MR |
|-------|---------------|-----------------|-----------------|-----------------|
| 1     | 63.0          | 94.0            | 79.0            | 125.0           |
| 2     | 115.0         | 172.0           | 141.0           | 234.0           |
| 3     | 125.0         | 204.0           | 234.0           | 266.0           |
| 4     | 172.0         | 282.0           | 266.0           | 406.0           |
| 5     | 203.0         | 359.0           | 360.0           | 531.0           |
| 6     | 250.0         | 406.0           | 484.0           | 484.0           |
| 7     | 297.0         | 469.0           | 546.0           | 579.0           |

## 6. 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] to demonstrate its scalability in terms of the number of members and under various network conditions.

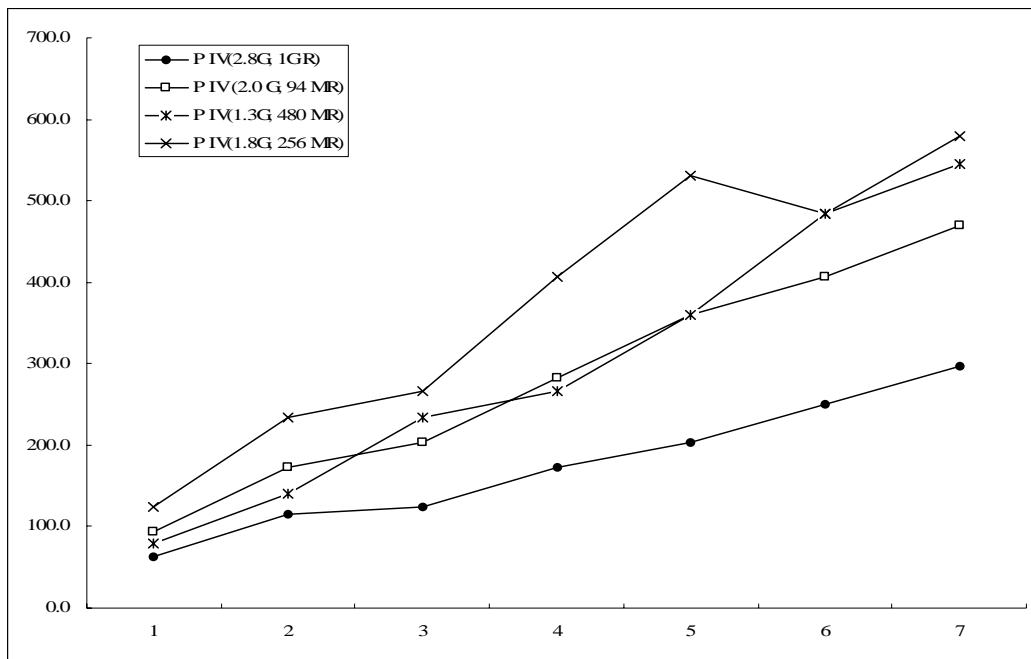


Figure 3. Group Key Computational Time for Each Machine

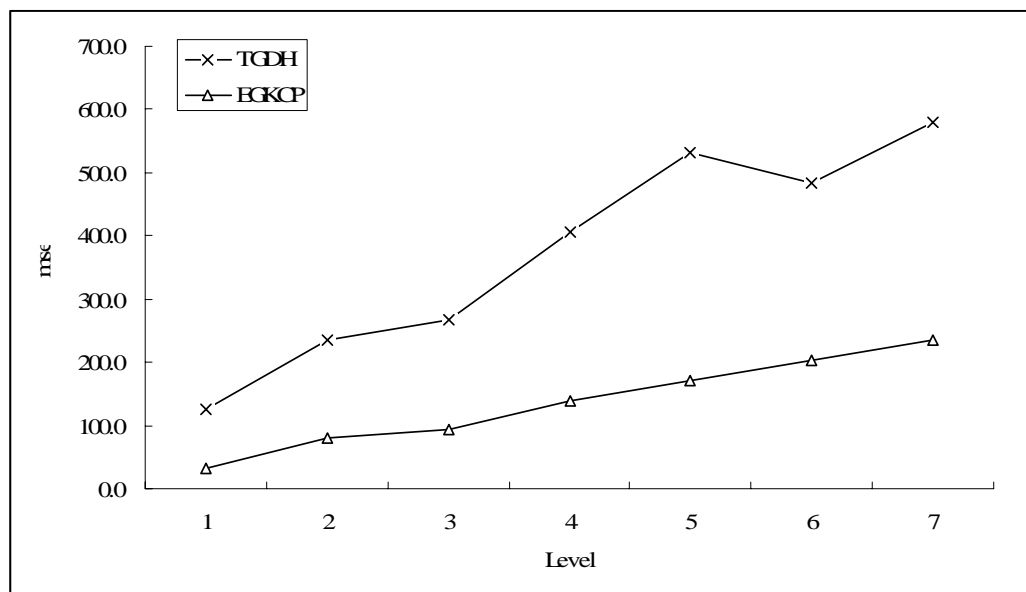


Figure 4. Performance Comparison with TGDH and EGKCP

## 7. References

- [1] Y. Kim, A. Perrig, and G. Tsudik, "Communication-efficient group key agreement", In 17th International Information Security Conference (IFIP SEC'01), June 2001.
- [2] Y. Kim, A. Perrig, and G. Tsudik, "Tree-based group key agreement", ACM Transaction on Information and System Security, 2004.
- [3] C. Wong, M. Gouda, and S. Lam, "Secure group communications using key graphs", IEEE / ACM Transactions on Networking, vol. 8, no. 1, Feb. 2000.
- [4] W. Diffie and M. E. Hellman. "New directions in cryptography", Transactions on Information Theory, IT-vol. 22, no. 6, pp. 644-654. Nov. 1976.
- [5] 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.
- [6] D. Wallner, E. Harder, and R. Agee, "Key management for multicast: Issues and architecture", Internet-Draft draft-wallner-keyarch-00.txt, June 1997.
- [7] 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, April 2004.
- [8] 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.
- [9] Y. Kim, "Group key agreement: theory and practice", Ph.D. thesis, May 2002.
- [10] E. Bresson, O. Chevassut, D. Pointcheval, and 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.
- [11] M. Steiner, G. Tsudik, and M. Waidner, "Cliques: A new approach to group key agreement", IEEE ICDCS'98, May 1998.
- [12] A. Fekete, N. Lynch, and A. Shvartsman, "Specifying and using a partitionable group communication service", ACM Transactions on Computer Systems, vol. 19, no. 2, May 2001.
- [13] 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, p. 457, 2004.
- [14] Donald Knuth. The Art of Computer Programming, Vol.1: Fundamental Algorithms, Third Edition. Addison-Wesley, pp. 107-123, 1997.
- [15] 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.
- [16] Ian Foster, The Grid: Blueprint for a New Computing Infrastructure, Second Edition, Elsevier, pp.47-53, 2004.