Secured Virtual Group Management in Grid

Poornaselvan KJ[†], Suresh Shanmugasundaram[††] and DivyaPreya Chidambaram[†††],

[†] **&** [††]Government College of Technology, Coimbatore, Tamilnadu, India
[†††]***Botho College, Gaborone, Botswana***

Summary
Grid components are constructed using internet as the backbone. Grid applications involve communication over these insecure open networks, thus security becomes an important requirement. The basic security while the VGs interact can be established only through secure and efficient group key management approaches. Centralized key management methods (key distribution) are apt for two-party (e.g., client-server or peer-to-peer) communication as well as for large multicast groups like the grid. However, many collaborative group settings (e.g., conferencing, white-boards, shared instruments, and command and-control systems and the VGs) require distributed key management techniques.
Key words:
Group Management, Key agreement, Grid Computing, Virtual Group Interaction in Grid, Security in Grid.

1. Introduction

1.1 Group Management Issues in Grid Ambience

The majority of protocols now available are mainly concerned with increasing the security while decreasing cryptographic computation costs. It has been long held as an unassailable fact that heavy weight computation such as large number arithmetic that forms the basis of many modern cryptographic algorithms which is the greatest burden imposed by security protocols. However, the continuing increase in computation power in the grid which uses modern workstations speeds up the heavyweight cryptographic operations and also the grid computations. For example, four years ago, a top-of-the-line RISC workstation performed a 512 bit modular exponentiation in around 24 ms. Four years later, an 850 MHz pentium III PC (priced at 1/5-th of the old RISC workstation) performs the same operation in less than 1 ms. In contrast; communication latency has not improved appreciably. The communication (especially via the Internet) has become both accessible and affordable which resulted in drastic increase in the demand for network bandwidth. While computation power and bandwidth are increasing, network delay has the lower bound dictated by the speed of light. More concretely, collaborative work groups like the VGs where the membersare dispersed across continents would expect considerable communication latency and thus benefit from protocols that minimize communication rounds. In this chapter a ground-breaking technique for group key agreement for the VGs and sub-VGs which is communication efficient is detailed.

1.2 Key Operations in VGS and Sub-VGS

1.2.1 VG Membership Operations

As the VGs and Sub-VGs constantly interact with the three primary components of the grid which are the users, providers and the brokers, in the internet, this calls for an approach which provides secured means of communication. Individual nodes and cluster of nodes can converse in this network. The reliability thus is a question and has to be resolved. As the previous chapters details the various operations of a node like joining a VG and leaving a VG dynamically which is not an unusual scenario in the grid ambience, a comprehensive approach that handles the group key agreement subsequent to these VG and Sub-VG membership changes is required in this underlying group communication system. The following VG membership changes are considered:
1. Single member operations which include member join and leave the VG
2. Multiple member operations which include group merge and group partition in VG
Join occurs when a prospective member joins a virtual group. **Leave** occurs when a member leaves a virtual group. **Partition** occurs when a virtual group (VG) is split into smaller sub virtual groups(sub VGs). A partition in a VG can take place for several reasons, two of which are quite common:
1) **Network failure** which occurs when a network event causes disconnection within the group. Consequently, a group is split into fragments some of which are singletons while others (those that maintain mutual connectivity) are sub-groups.

Manuscript received June 5, 2010
Manuscript revised June 20, 2010

2) **Explicit (application-driven) partition** which occurs when the application decides to split the group or simply exclude multiple members at once.
Merge occurs when two or more groups merge to form a single group (a group merge may be voluntary or involuntary):
1) Network fault heal occurs when a network event causes previously disconnected network partitions to reconnect.
2) Explicit (application-driven) merge occurs when the application decides to merge multiple pre-existing groups into a single
group.
In practice, however, such events are common owing to network configuration and router failures.
Undeniable arguments in support of these claims are available (Moser et al, 1994). Hence, dealing with partitions and merging in
virtual groups is a vital component of group key agreement.

1.2.2. Cryptographic Attributes in a VG

The desired properties for a secure virtual group key agreement protocol for special networking ambience
like the grid (Kim et al, 2000) are as follows.
• Virtual Group Key Secrecy guarantees that it is computationally infeasible for a passive adversary to discover any
virtual group key.
• Forward Secrecy guarantees that a passive adversary who knows a contiguous subset of virtual group keys cannot
discover subsequent virtual group keys.
• Backward Secrecy guarantees that a passive adversary who knows a contiguous subset of group keys cannot
discover preceding virtual group keys.
• Key Independence guarantees that a passive adversary who knows any proper sub-VG keys cannot discover any other VG key not included in the subset. Backward and Forward Secrecy attributes assume that the adversary is a current or a former Virtual Group member. The other attributes additionally include the cases of unintentionally leaked or otherwise compromised Virtual Group keys.

2. Protocol Variants for VG Key Agreement

2.1 Skinny Tree Protocol (STRP)

The existing skinny tree (STR) protocol uses Diffie-Hellman key exchange technique with imbalanced key
tree. The degree of communication efficiency is higher when compared to Centralized Key Distribution Protocol, Burmester-Desmedt Protocol, Group Diffie-Hellman Protocol, Skinny Tree Protocol. The security of the Diffie-Hellman key exchange lies in the fact that, while it is relatively easy to calculate exponentials modulo a prime, it is very difficult to calculate discrete logarithms. For large primes, the latter computational task is considered infeasible. The downside is that it is computational
intensive as it involves exponentials. This may not be feasible to apply in the VG communications as grid computations are also intensive and becomes more rigorous when accomplished with the above mentioned discrete logarithmic computations. Hence an approach for enabling secured VG communication through an efficient key agreement is required. This is possible by Elliptic Curve Crypto Systems(ECCS) which is detailed in the next sub chapter.

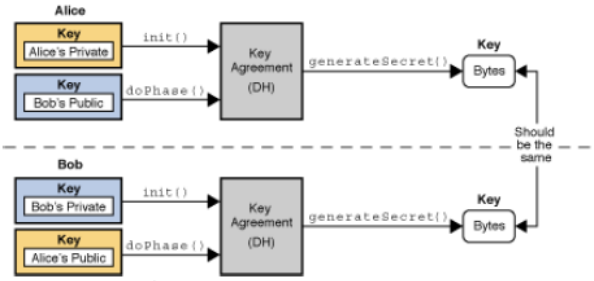

Fig 5.4: Group Key Agreement via STRP

3. Elliptic Curve Cryptosystems

Elliptic Curve Cryptosystems (ECC) were proposed independently in 1985 by Victor Miller and Neal Koblitz. At the time, both Miller and Koblitz regarded the concept of ECC as mathematically elegant; however felt that its implementation would be impractical. Since 1985, ECC has received intense scrutiny from cryptographers, mathematicians, and computer scientists around the world. These advantages are particularly beneficial in applications where bandwidth, processing capacity, power availability or storage is constrained like the Grid projects.

3.1 Enhanced VG Key Agreement Approach

The group key agreement helps for secure group communication over insecure open networks, by establishing a unique group key to communicate.
The key setup latency is influenced by communication rounds and cryptographic rounds. However continuous
advancements in computation power where not matched with decrease in communication delay. Even though computation power and bandwidth are increasing, network delay still has the lower bound dictated by the speed of light. Hence an efficient protocol that reduces the

communication rounds could benefit the collaborative computing society.

3.2. Rudiments and Assumptions of the Approach

The approach which is detailed here uses an imbalanced tree coupled with Elliptic curve Diffie-Hellman key exchange (ECDH). Here two-party ECDH is extended for a Virtual Group. This tree has two types of nodes: leaf and parent as explained in the previous chapter. Each leaf node is associated with a specific Virtual Group. An internal node IN<i> always has two children: another (lower) internal node IN<i-1> and a leaf node LN<i>. The exception is IN<i> which is also a leaf node corresponding to M1. (Note that consequently, r1=k1.)

3.2.1 Basic Key Agreement Protocol

A few assumptions that are made before proceeding are detailed below. All members know the structure of the key tree and their initial position within the tree at any instant of time as the number of VGs are known at any instant period of time. Furthermore, each node in the VG knows its session chance and the blinded session chance of all other VGs or sub-VGs. The two members M1 and M2 can first compute the Virtual group key corresponding to IN<2>. Key exchange using elliptic curves in the Virtual Groups can be done in the following manner:

1. Pick a large integer q, which is either a prime number p or an integer of the form 2m and elliptic curve parameters a, b for equation of the form, y2 mod p = (x3+ax+b) mod p. This defines the elliptic group of points Eq (a, b).
2. Pick a base point G=(x1, y1) in Ep (a, b).) whose order is very large n.
3. A key exchange between Virtual Groups VG-A and VG-B can be accomplished as follows:

a. VG-A selects an integer less than n .This is VG-A's private key. VG-A then generates a public key PA =n A * G; the public key is a point in Eq (a,b).
b. VG-B similarly selects a private key n B and computes a public key PB.
c. Both the VGs exchange their public keys.
d. VG-A generates the secret key K = n A * PB and VG-B also generates the secret key K = n B * PA.

For elliptic curve cryptography, an operation over elliptic curves, called addition, is used. Multiplication is defined by repeated addition. For example, a * k = (a + a + ……+ a) [k times] where the addition is performed over an Elliptic curve.

The addition operation is as follows:

If P=(x p , y p) and Q=(x Q , y Q) with P _ –Q , then R=P+Q=(x R , y R) is determined by the following rules:

x R =(_2 – x p – x Q) mod p. y R =(_ (x p – x Q) – y p) mod p. where _ = ((y Q-y p)/(x Q -y p)) mod p if P _ Q. _ = ((3x2 p +a)/(2y p)) mod p if P = Q.

Note that the secret key is pair of numbers. If this key is to be used as a session key for conventional encryption between the VGs, then a single number must be generated. x coordinates or some simple function of the x coordinates could be used. Finally the bkeys for the above secret key K by using same formula as used in public key calculation such as bki = (K * G) mod p is calculated. If any member joins this Virtual Group, then by exchanging their public keys the new node and the group can calculate the key for that Virtual Group. The secret key Ki (i>1) is a result of an ECDH key exchange between the nodes of the VG (K1 is an exception which is equal to r1) which can be computed recursively as follows. Ki=(bki-1)*ri mod p = (bri) * Ki-1 mod p if i>1 , bri=(ri*G)3. The root (group) key is never used directly for the purposes of encryption, authentication or integrity. Instead, such special-purpose sub-keys are derived from the root key, e.g., by applying a cryptographically secure hash function to the root key. All bkeys are assumed to be public.

The key tree is shown in figure 5. The two members M1 and M2 can first compute the group key corresponding to IN<2>. M1 computes: k2=(br2)* r1 mod p=(r1*r2*G) mod p; bk2= (k2*G) mod p

k3 = (br3)*k2 mod p; bk3 = (k3*G) mod p

.

.

kn = (brn)*kn-1 mod p

Next, M1 broadcasts all bkeys bki with 1 < i < N - 1. Armed with this message, every member then computes kn as follows. (As mentioned above, members M1 and M2 derive the group key without additional broadcasts.) Any Mi (with i > 2) knows its session random ri and bki-1 from the broadcast message. Hence, it can derive ki = (bk i-1)* ri. It can then compute all remaining keys recursively up to the group key from the public blinded session random keys : (bri)* ki-1 mod p i<=n The protocols that make up the proposed virtual group key management suite: join, leave, merge and partition share a common framework with the following features:

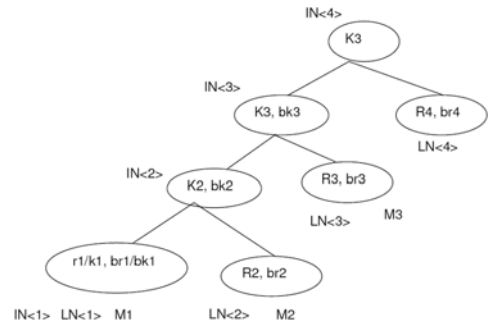

Figure 5. Notation of Key tree

Each Virtual Group member contributes an equal share to the Virtual Group key; this share is kept secret by each Virtual Group member. The Virtual Group key is computed as a function of all current Virtual Group members' shares. As the Virtual Group grows, new members' shares are factored into the Virtual Group key while the remaining members' shares (except for sponsor who changes its session random to provide key independence) stay unchanged. As the Virtual Group shrinks, departing members' shares are removed from the Virtual Group key and at least one remaining member changes its share. In a join or a merge, sponsor is associated with the topmost leaf node of each key tree. In a leave or a partition, sponsor is located immediately below the deepest leaving node.

3.2.2. Join

In a virtual group having n nodes {M1, M2,…, Mn), the group communication system announces the arrival of a new member. Both the new member and the prior virtual group members receive this notification simultaneously. The new member Mn+1 broadcast a join request message that contains its own bkey bkn+1 (which is same as its session random br n+1). Upon receiving this message, the current virtual group's sponsor Mn refreshes its session random, computes brn,kn,bkn, and sends the current tree BT<n> to M n+1 with all bkeys. Next, each member Mi increments n and creates a new root key node IN<n> with two children: the root node IN<n-1> of the prior tree Ti on the left and the new leaf node LN<n> corresponding to the new member on the right. Note that every member can compute the virtual group key since: All existing members only need the new member's blinded session random. The new member needs the blinded virtual group key of the prior virtual group. In a 'join' operation, the sponsor is always the top most leaf node, i.e., the most recent member in the current virtual group. Figure 5.6 shows an example of a new member Ms joining a virtual group. To provide forward secrecy, the sponsor M4 updates its session random r4. And as described 'join' takes two communication rounds and five cryptographic operations to compute the new virtual group key (four by the sponsor and two by everyone else). In a virtual group of n members when a member Md (d _ n) leaves the group and if d>1, the sponsor Ms is the leaf node and directly a member, i.e., Md-1. Otherwise, the sponsor is M2. Upon hearing about the leave event from the group communication system, each remaining member updates its key tree by deleting the nodes LN<d> corresponding to Md and its parent node IN<d>. The nodes above the leaving node are also renumbered. The former sibling IN<d-1> of Md is promoted to replace (former) Md's parent. The sponsor Ms selects a new secret session random, computes all keys (and bkeys) just below the root node, and broadcasts BT<s> to the group. This information allows all members (including the sponsor) to recomputed the new virtual group key. Figure.3.3 describes the Leave protocol.

Figure 5.7 show that if M4 leaves the virtual group, other members delete the leaving node along with its parent. Then, the sponsor M3 refreshes session r3, computes br3', k3', bk3', and broadcasts the updated tree BT<4>. Upon receiving the broadcast, all members (including M3) compute the virtual group key K4. Note that M4 cannot compute the virtual group key (even though it knows all bkeys) since its session random is no longer part thereof.

The leave protocol takes one communication round and involves a single broadcast. The cryptographic cost varies depending upon two factors:

1) The position of the departed member in a VG

2) The position of the remaining needing to compute the new key.

The total number of serial cryptographic operations in the leave protocol can be expressed as:

2(n-d) +1+ (n-d) +1=3n-3d+2 when d>2 3n-7 when d=1, 2

In the worst case, M1, M2 or M3 leaves the virtual group. The cost for this leave operation is equal to 3n-7. The expected leave cost is 3(n/2) +2.

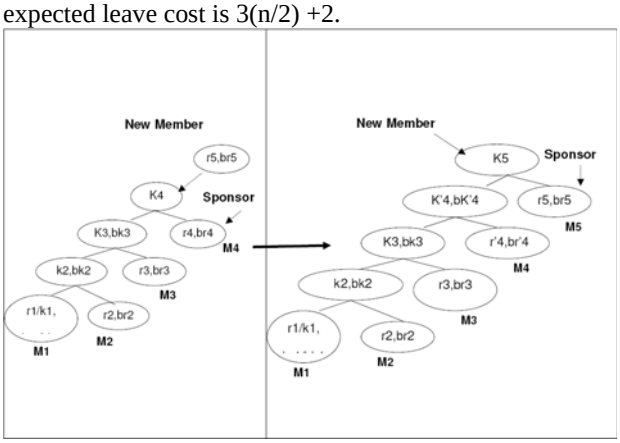

Figure 6: Tree updation in Join

3.2.3 Leave

The Leave protocol provides forward secrecy since a former member cannot compute the new key owing to the sponsor's changing the session random.

3.2.4 Partition

A network fault (or severe congestion) can cause a partition of the virtual group. To the remaining members, this actually appears as a concurrent leave of multiple nodes. With a minor modification, the 'Leave' protocol can handle multiple nodes leaving the VG in a single round. The only difference is in sponsor selection. In case of a partition, the sponsor is the leaf node direct below the lowest-numbered leaving member. (If M1 is the lowest-leaving member the sponsor is the lowestnumbered surviving member.) After deleting all leaving nodes, the sponsor Ms refreshes its session random (key share), computes keys and bkeys going up the tree as in the plain leave protocol. It then broadcasts the updated key tree BT<s> containing only blinded values. Each member (including Ms) can now compute the group key. Figure 5.8 shows an example where the sponsor deletes all nodes of leaving members and computes all necessary keys and bkeys in the first round. In this example, M1 is the sponsor since M2 left the virtual group. After picking a new session random r1 the sponsor computes K2 and bk2, and broadcasts the whole tree. Upon receiving this message, every member can compute the new virtual group key k3. Note that session random and blinded session random are renumbered as in the leave protocol. The computation and communication complexity of this protocol is identical to that of the leave protocol. The same holds for its security properties.

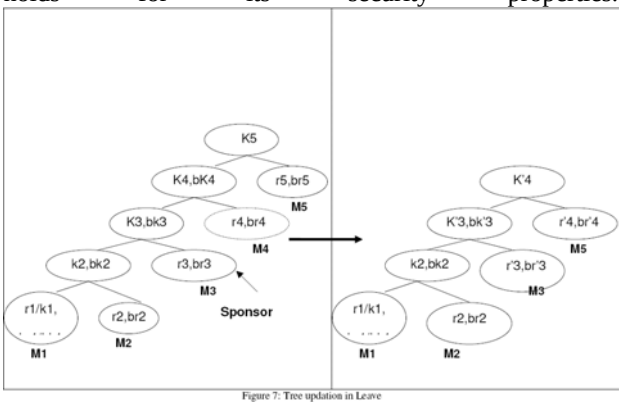

Figure 7: Tree updation in Leave

In Merge protocol, as in the join case, the communication system simultaneously notifies all virtual group nodes (in all groups) about the merge event. Moreover, reliable virtual group communication toolkits typically include a list of all members that are about to merge in the merge notification. More specifically, it requires that each member be able to distinguish the virtual group it was in from the virtual group that it is merging with. It is natural to graft the smaller tree atop the larger tree. If any two trees are of the same height, then unambiguous ordering is used. When merging two trees, the lowest-numbered leaf of the smaller tree becomes the right child of a new intermediate node. The left child of the new intermediate node becomes the root of the larger tree. Using this technique recursively, multiple k-ary trees are merged as shown in Figure 5.9.In the first round of the merge protocol, all sponsors (members associated with topmost leaf node in each tree) exchange their respective key trees containing all blinded session random. The highest-numbered member of the largest tree becomes the sponsor of the second round in the merge protocol. After refreshing the session random, this sponsor computes every (key, bkey) pair up to the intermediate node just below the root node using the blinded session random of the other virtual group members. The sponsor then broadcasts the key tree with the bkeys and blinded session random to the other members. All members now have the complete set of bkeys, which allows them to compute the new virtual group key.r1/k1 ,

3.2.5 Merge

Figure 5.9 shows an example of merging two trees. After the merge notification, the sponsors M4 and M7 broadcast their key trees containing all blinded session random. Upon receiving these broadcast messages, every member in both virtual groups reconstructs the key tree. The smaller tree with three members is placed on top of large tree with four members. Every member generates a new intermediate node IN<5> and makes it the parent of the old root node IN<4>of the larger tree and the previous leftmost leaf node LN<5>. Both intermediate nodes 'IN<1>' and 'IN<2>' of the previous smaller tree has then need to be renumbered as IN<6>and IN<7> respectively. The new intermediate node IN<5> also becomes the child of the previous lowest intermediate node IN<6>. Using the previous blinded group key at IN<4> of the larger group and blinded session random br5 and br6, the sponsor in the second round, M4, computes all intermediate keys and bkeys (k4; bk4; k5; bk5; k6; bk6) except the root node. Finally, it broadcasts BT<4> that contains all bkeys and blinded session random keys up to IN<6>. Upon receipt of the broadcast, every member can compute the virtual group key. In summary, the merge protocol runs in two communication rounds.

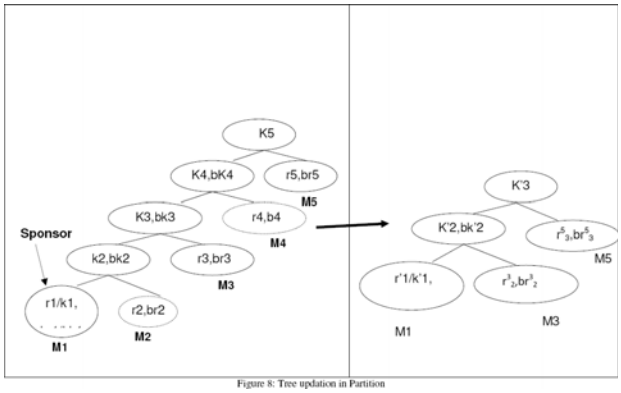

Figure 8: Tree updation in Partition

4. Modules

There are four main modules for accomplishing secure communication for VG's interaction. The descriptions of each of the modules are detailed in the following sub-chapters.

4.1. JOIN

This module is responsible for generating a new virtual group key whenever a new member joins the virtual group.
Procedure: Step 1: The new member or node broadcasts request for join Step 2: Every member • Updates key tree by adding new member node and new root node,
• Removes bkn, The sponsor Mn additionally
• Generates new share rn and computes brn, kn, bkn
• Broadcasts updated tree BT<n>.
Step 3: Every member computes the virtual group key Kn+1 using BT<n>.

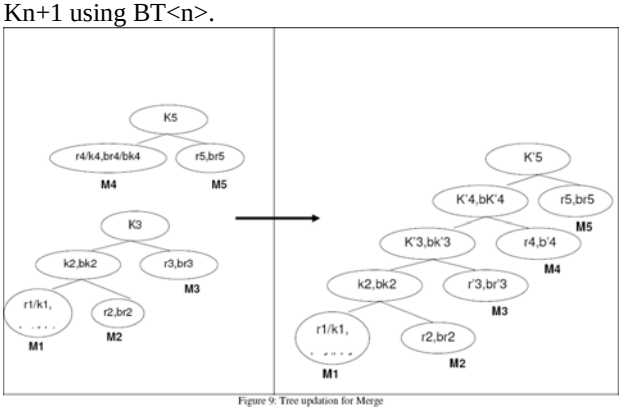

Figure 9: Tree updation for Merge

4.2 Leave

This module generates a new virtual group key removing the leaving member's node from the VG.
Procedure: Step1: Leaving member sends leave notification to the virtual group. Step2: Every member updates key tree by removing the leaving member's node and renumbering the nodes above it. The sponsor Mn additionally, Generates new share rn and computes all keys and bkeys up to the last member. Broadcasts the updated key tree BT<s>to all its members.Step 3: Every member generates new virtual group key using BT<s>.

4.3 MERGE

This module generates new virtual group key whenever two virtual groups merge in to a single virtual group.
Procedure: Step 1: Each sponsor Msi in Tsi for *i* _ [1,k] Broadcasts tree BT<si> Step 2: Every Member
Updates key tree by merging all trees, Removes all keys and bkeys from the sponsor node, The sponsor Ms (additionally) , Generates new share rs and computes brs.
• Computes all keys and bkeys from its parent to the node just below root, Broadcasts updated tree BT<s> .
Step 3: Every member computes the virtual group key using BT<s> .

4.4 PARTITION

This module generates a new virtual group key by identifying the subsequent leave of multiple members.
Procedure:
Step 1: In a virtual group of members leaving subsequently identify the lowest leaving member. The member below
him is the sponsor if i>1, else it is the least surviving member.
Step2: Every surviving member updates key tree by removing the leaving members' nodes and renumbering the
nodes.
The sponsor Mn additionally, Generates new share rn and computes all keys and bkeys up to the last member and broadcasts the updated key tree BT<s>to all its members.
Step 3: Every member generates new virtual group key using BT<s>

5. Experimentation and Outcome

This section details the working environment of the approach that can be applied in the VGs. It also describes some details about various classes used for this technique.
The programming language used to implement and study this process is Java. The codes were developed using JBuilder. The operating system in which this implementation was carried out is Windows XP. And the back end used was Oracle 8i. The hardware environment is the same which is explained in chapter 3.

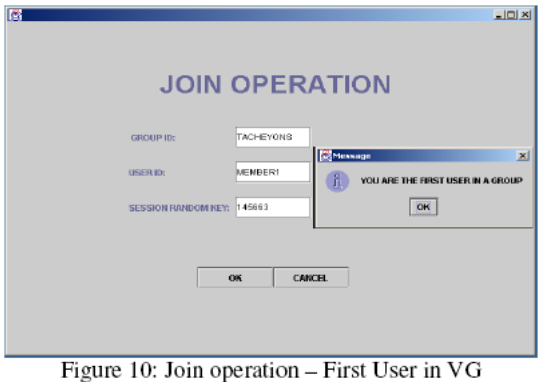

Figure 10: Join operation – First User in VG

5.1 Protocol Pseudo code

A simplified version of the code in the pseudo form is listed below for the experimentation. The key operations like key generation, join, leave and merge in the VGs are the primarily focused. This pseudo code is extended to a java project to further exemplify.

VGs/sub-VGs receive message (message type = membership event)
Construct new tree
While there are missing bkeys {
If ((Member computes missing keys and member is the sponsor) || (sponsor computed a key))
While (true) {
Compute missing (key, bkey) pairs
If (Member cannot compute)
Break
End if
If (Members require information)
Broadcast new bkeys
End if
End if
Receive message
If (message type = membership event)
Construct new tree
End if
End while } }

Various classes and methods were declared and defined for the experimentation and the descriptions are detailed below.

The different classes used are Join, JoinKeyGen, GroupChat, LeaveKeyGen, Merge, and EllipticCurve.

5.1.1 Join Class

Join class uses the the following methods;

_ Firstuser()
_ Nextuser()

This class uses the Firstuser() method to check whether the user joined is first user or not. If the user is the first user in initiating the group, the user is allowed to wait for connection. For other than first user Nextuser() method is called. The Nextuser() method then gets the new members random key and computes new set of group keys after the sponsor refreshes the session random key by invoking the JoinKeyGen class.

5.1.2 JoinKeyGen Class

JoinKeyGen class as the name states, is used for key generation in a VG. For implementing this, JoinKeyGen class uses the methods listed below;

_ keyGen()
_ sendUpdate()

The keyGen() method is used to generate new set of group keys after every join membership operation in a VG or its sub-VG. It uses sendUpdate() method to transmit the updated key to the new and existing users.

5.1.3 GroupChat

GroupChat class is used to exemplify VG interactions and the methods used are chat(), sponsor(), run() and partition(). The chat() method is used for sending and receiving messages among the virtual groups. It also checks whether the received message is a control message or a data message. If it is a control message then it invokes the appropriate method and enables the sponsor for key refresh. If it is a data message then it displays in the chat application. The sponsor() method gets the new session random key from the corresponding user and generates the new set of keys for the group. It then transmits the updated key to the existing members. The run() method invokes the thread which simultaneously handles sending and receiving of messages. The partition() method listens for the subsequent leaving of a number of users. It invokes the LeaveKey class for generating new set of keys similar to leave operation.

5.1.4 LeaveKey Class

LeaveKey class has the methods which supports the leave operation. The methods are leave() and leaveKeyGen() which are invoked appropriately to accomplish leave operation. The leave() method on getting the leave notification from the leaving member, identifies the sponsor and calls the sponsor() method. The leaveKeyGen() method on getting the leave_reply message from the sponsor, generates the new set of virtual group key for it using the updated bkey in the message.

5.1.5 Merge

Merge class takes care of the merging of the virtual groups and sub-groups. This is achieved by the mergeKeyGen()

method and the mergeUpdate() method. This class is invoked when there is a need to merge the user's virtual group to another virtual group. This applies to the sub VGs also. The mergeKeyGen() method is called whenever a merge_request message hits the socket. Later it invokes the sponsor() method if this member is the sponsor. The mergeUpdate() method sends merge_reply message to the virtual group nodes which contains the sponsor refreshed key.

5.1.6 EllipticCurve Package

This is a user defined package that is used to generate blinded group key and blinded random key by getting the virtual group key and random key respectively.

5.2 Protocol Format

All protocol messages which is used for the VG interaction include the following attributes: **Sender information**: name of the sender. **Group information**: unique name of the group. **Membership information**: names (and other information) of current group members. **Message type**: unique message identifier for each protocol message.
Key epoch: strictly increasing counter. Whenever there is a new membership event occur, each member increments the key epoch. If two virtual groups VG1 and VG2 merge, the resulting epoch is: epochnew = max (epochG1 ; epochG2) + 1. Key epoch is the same across all current virtual group nodes. If a virtual group member receives a protocol message with a smaller than current epoch, it terminates the protocol assuming suspected replay.

5.3 Exemplifications of VG Operations

Figure 5.10 shows first member joining the group giving the group ID user ID his session random key. A message box is displayed to tell him the user is the one initiated the connection.

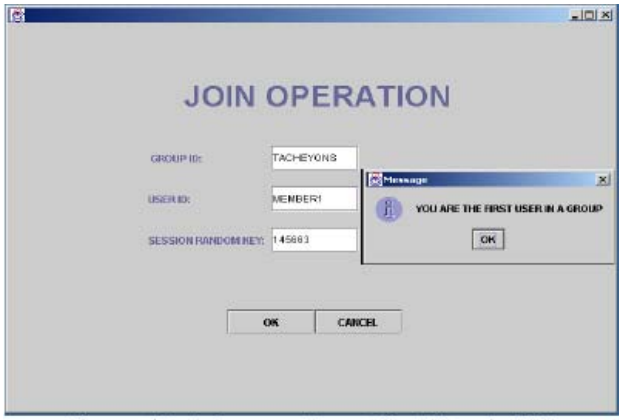

Figure 10: Join operation – First User in VG

Figure 11 shows Wait for Connection frame that is enabled until the first user gets connection from any other node
interested in the virtual group.

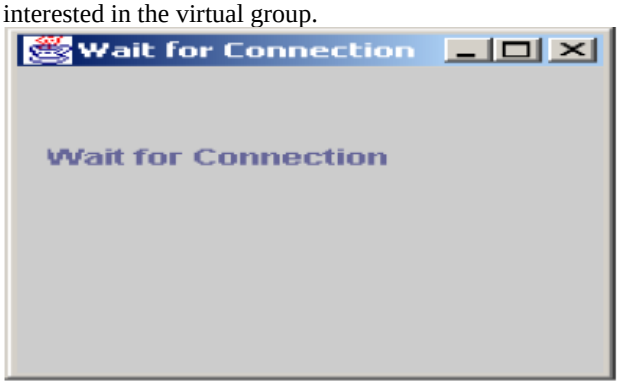

Figure 11: Wait for connection

Figure 12 shows the Join notification send by new user to the existing users on joining the virtual group.

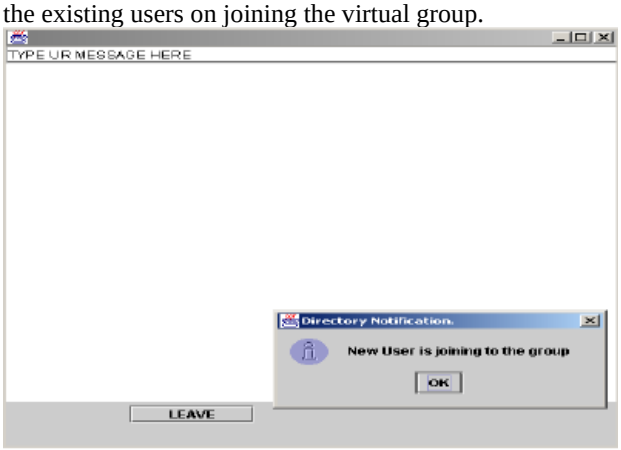

Figure 12: Join Notification from New User

Figure 13 shows the Sponsor refreshing the random key on every membership change operation like Join, Leave, Merge, and Partition that creates consequent impact on the VGs and sub-VGs

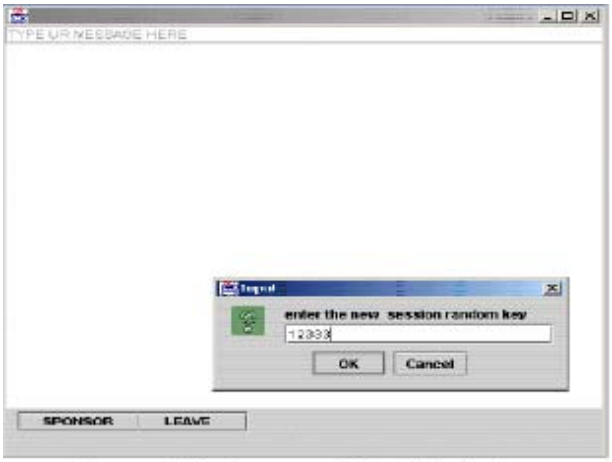

Figure 13: Sponsor Key Updation

Figure 14 shows the new group key established within the group after the sponsor had refreshed the session random key.

Figure 14: New Updated Virtual Group key

Figure 15 shows the Virtual Group members chatting after establishing the new virtual group key.

Figure 5.15: Chat form

Figure 5.16 shows the leave notification send to the rest of the members when the Member 2 in the virtual group leaves.

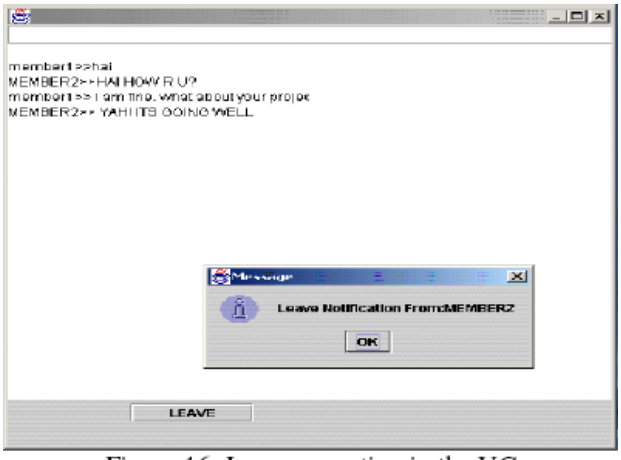

Figure 16: Leave operation in the VG

Figure 17. shows a partition as two members subsequently send a leave notification.

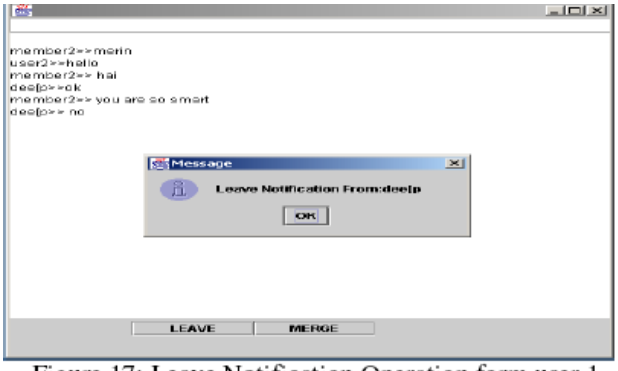

Figure 17: Leave Notification Operation form user 1

Figure 17: Leave Notification Operation form user 1

5. Conclusion and Future Enhancements

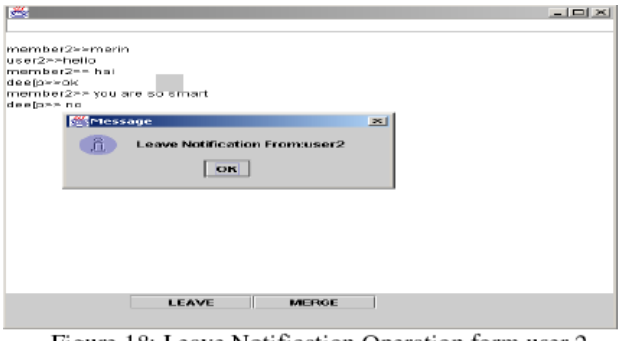

Figure 18: Leave Notification Operation form user 2

Figure 18: Leave Notification Operation form user 2

Figure 19. shows a merge that is users of current group named Xplore willing to merge with another group named tacheyons.

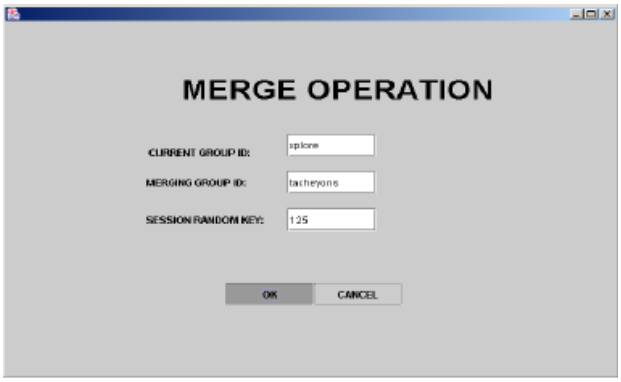

Figure 19: Merge operation

6 Conclusions and Future Enhancement

A Secure contributory virtual group key agreement approach (Modified STR) which is applied in the grid framework for VG interaction is detailed in this chapter. Modified STR supports all dynamic peer group operations which is very common in Grid computations and interactions. The key operations are Join, Leave, Merge, and Partition. Furthermore, it easily handles cascaded (nested) VG membership events and network failures.

It has been an assumption that the computation power of processors increases and thus reducing the computation costs incurred in key setup. Eventually communication costs which a lower bound has dictated by the speed of light. The proposed protocol is already the most efficient group key agreement protocol over high-delay wide-area networks; it will become more advantageous as processor speeds increase. Effective virtual group key generation is implemented and detailed in this chapter. Using this key for the VG, sub keys for sub VGs can be derived e.g., by applying a cryptographically secure hash function.

This approach has been exemplified by the group chat application in the VG and sub-VGs. In future this can be extended to cloud computing as well.

REFERENCES

[1] A.Foster and C. Kesselman (editors), The Grid: Blueprint for a Future Computing Infrastructure, Morgan Kaufmann Publishers, USA, 1999.

[2] A.Sulistio and R. Buyya. A time optimization algorithm for scheduling bag-of-task applications in auction-based proportional share systems. In Proceedings of the 17th International Symposium on Computer Architecture and High Performance Computing (SBAC-PAD), Rio de Janeiro, Brazil, October 2005.

[3] Ahmar Abbas, 'Grid Computide to Technology and Appling: A Practical Guide to Technology & Applications',Firewall Media, New Delhi. India, 2009

[4] Anthony Sulistio, Gokul Poduval, Rajkumar Buyya, and Chen-Khong Tham, 'Constructing A Grid Simulation with Differentiated Network Service Using GridSim', A. Oram (editor), Peer-to-Peer: Harnessing the Power of Disruptive Technologies, O'Reilly Press, USA, 2001.

[5] B. Allcock, I. Foster, V. Nefedova, A. Chervenak, E. Deelman, C. Kesselman, J. Lee, A. Sim, A. Shoshani, B. Drach and D. Williams, High-Performance Remote Access to Climate Simulation Data: A Challenge Problem for Data Grid Technologies, Proceedings of SC2001 Conference, Denver, USA, November 2001.

[6] B. Knighten, Peer to Peer Computing Working Group, Intel Developer's Forum, August 24, 2009, ttp://www.peer-topeerwg.org

[7] B. Krishnamurthy and J. Wang, "Topology Modeling via Cluster Graphs," Proceedings of the First ACM SIGCOMM Internet Measurement Workshop: IMW 2001, pp. 19-23, 2001.

[8] C. A. Waldspurger, T. Hogg, B. A. Huberman, J. O. kephart, and S. Stornetta, "Spawn: A distributed computational economy," IEEE Transactions on Software Engineering, vol. 18, pp. 103-177, 1992

[9] C. Dumitrescu and I. Foster. Usage policy-based cpu sharing in virtual organizations. In Proceedings of the 5th IEEE/ACM International Workshop on Grid Computing (GRID 2004), Pittsburgh, USA, November 2004

[10] C. L. Dumitrescu and I. Foster. GRUBER: A Grid resource usage SLA broker. In Proceedings of the 11th International European Parallel Computing Conference (EuroPar), Lisbon, Portugal, 2005.

[11] Cantu-Paz E, Designing Efficient and Accurate Parallel Genetic Algorithms, Technical Report No. 99017, Illinois Genetic Algorithms Laboratory, UIUC, USA, July 1999.

[12] D. Abramson, J. Giddy, and L. Kotler, High Performance Parametric Modeling with Nimrod/G: Killer Application for the Global Grid?, Proceedings of the International Parallel and Distributed Processing Symposium (IPDPS 2000), May 1-5, 2000, Cancun, Mexico, IEEE Computer Society (CS) Press, USA, 2000.

[13] D. Ferguson, C. Nikolaou, J. Sairamesh, and Y. Yemini, Economic Models for Allocating Resources in Computer Systems, In Market-based Control: A Paradigm for Distributed Resource Allocation, World Scientific Press, Singapore, 1996.

[14] D. Kondo, H. Casanova, E. Wing and F. Berman (2002), "Models and Scheduling guidelines for Global Computing Applications", Proceedings of International Parallel and Distributed Processing Symposium, pp 437 – 443.

[15] D. Morrison, 'PATRICIA—Practical Algorithm To Retrieve Information Coded In Alphanumeric (Oct. 1968)' J.ACM, vol. 15, no. 4, pp. 514–534.

[16] E. Elmroth and P. Gardfjall. Design and evaluation of a decentralized system for Grid-wide fairshare scheduling. In Proceedings of 1st IEEE Conference on e-Science and Grid Computing, Melbourne, Australia, December 2005.

[17] Emmanuel Bresson, Olivier Chevassut, David Pointcheval, and Jean-Jacques Quisquater. Provably authenticatedgroupdiffie-hellman key exchange. In Pierangela Samarati, editor, 8th ACM Conference on Computer and Communications Security, Philadelphia, PA, USA, November 2001. ACM Press.

[18] F. Zhang, S. Liu and K. Kim, ID-Based One Round Authenticated Tripartite Key Agreement Protocol with Pairings,"Available at http://eprint.iacr.org, 2002.
[19] G. Cordasco, G. Malewicz, A.L. Rosenberg (2006): Advances in a dag-scheduling theory for Internet-based computing. See also, On scheduling expansive and reductive dags for Internet-based computing. 26th Intl. Conf. on Distributed Computing Systems, 2006.
[20] G. Malewicz, A.L. Rosenberg, M. Yurkewych (2006): Toward a theory for scheduling dags in Internet-based computing. IEEE Trans. Comput. 55, 757–768.
[21] Gennaro Cordasco, Grzegorz Malewicz and Arnold L. Rosenberg Applying IC-Scheduling Theory to Familiar Classes of Computations, 2006
[22] Gnutella - http://gnutella.wego.com/
[23] Grid Computing Information Centre: http://www.gridcomputing.com.
[24] Grid Forum – http://www.gridforum.org
[25] Herbert Shield (2005), "Java: The complete Reference, J2SE 5 Edition", Tata McGraw-Hill Edition.
[26] J. In, P. Avery, R. Cavanaugh, and S. Ranka. Policy based scheduling for simple quality of service in Grid computing.In Proceedings of the 18th Annual International Parallel and Distributed Processing Symposium (IPDPS), Santa Fe,USA, April 2004.
[27] J. Nakai, "Pricing Computing Resources: Reading Between the Lines and Beyond," Technical Report NAS-01-010,2002.
[28] Joshy Joseph, Craig Fellenstein (2004), "Grid Computing", Pearson Education Publications
[29] K. Holtman, CMS Data Grid System Overview and Requirements, The Compact Muon Solenoid (CMS) Experiment Note 2001/037, CERN, Switzerland, 2001.
[30] K. Reynolds, The Double Auction, Agorics, Inc., 1996.http://www.agorics.com/Library/Auctions/auction6.ht
[31] K.J.Poornaselvan, S.Suresh, 'Incentive-based Scheduling Framework for Grid Computing', International Conference on Software Engineering and Applications (SEA 2007), at Cambridge, MA, USA during November 19-21, 2007
[32] K.J.Poornaselvan, S.Suresh, 'Spur Policy for Income Administration in Grid Computing', proceedings of the IASTED Conference , MSAfrica 2008 page no 164-169 ,hosted by University of Botswana, on 7-10, September, 2008.
[33] K.J.Poornaselvan, S.Suresh, C.G.Gayathri and Ankit Kamal Mehta, Gridlock Avoidance Scheduling for Tree Structured Computation, Proceedings of National Conference on Advanced Computing 2007, pp 71-81, Department of Computer Science & Engineering, Anna University, MIT Campus, February 16-17, 2007, Chennai, India.
[34] K.J.Poornaselvan, S.Suresh, P.Birahadish, A.Saravanan ,'Enhanced Scheme for Group Key Agreement', Proceedings of the International Conference on Digital Communications & Computer Applications (DCCA 2007),pp 242-253 at Jordan University of Science & Technology, Jordan during March 19-22, 2007.
[35] K.J.Poornaselvan, S.Suresh, P.Birahadish, A.Saravanan, 'Modified Skinny Tree Group Agreement Protocol',Proceedings of National Conference on Advanced Computing 2007, pp 160-178, Department of Computer Science & Engineering, Anna University, MIT Campus, February 16-17, 2007, Chennai, India.
[36] K.J.Poornaselvan, S.Suresh,'Gridlock Prevention by Job Split-up', International Journal of Computer Science & Network Security, Korea, Vol 9: No.03 March 2009 edition.
[37] K.J.Poornaselvan, Suresh S, C.DivyaPreya, C.G.Gayathri, 'Efficient IP Lookup Algorithm' , Special Topics in Computing and ICT Research- Strengthening the Role of ICT in Development, Volume III, ISBN 978-9970-02-730-9,Makerere University, Fountain Publishers, Kampala, Uganda, pp. 111-122, 2007.
[38] L. Gong, X. H. Sun, and E. Waston, "Performance modeling and prediction of non-dedicated network computing,"IEEE Trans. on Computer, vol. 51, pp. 1041-1055, 2002.
[39] L. Norskog, A Personal Communication on Economics and Grid Allocation, Enron Broadband Systems, March 14, 2001. Available at: http://www.buyya.com/ecogrid/comments/enron.txt
[40] L. W. McKnight and J. Boroumand, Pricing Internet Services: Approaches and Challenges, IEEE Computer, Vol. 33, No. 2, pp. 128-129, IEEE CS Press, USA, Feb. 2000.
[41] M. A. Ruiz-Sanchez, E.W. Biersack, and W. Dabbous (Mar.–Apr. 2001) 'Survey and taxonomy of IP address lookup algorithms,' IEEE Network, vol. 15, no. 2, pp. 8–23.
[42] M. Baker (editor), Grid Computing, IEEE DS Online, http://computer.org/dsonline/gc/
[43] M. Baker, R. Buyya, and D. Laforenza, The Grid: International Efforts in Global Computing, International Conference on Advances in Infrastructure for Electronic Business, Science, and Education on the Internet (SSGRR 2000), l'Aquila, Rome, Italy, July 31 - August 6. 2000.
[44] M. J. Gonzalez, "Deterministic Processor Scheduling," ACM Computing Surveys, vol. 9, pp. 173-204, 1997.
[45] M. Waldvogel, G. Varghese, J. Turner, and B. Plattner (Oct. 1997) 'Scalable high speed IP routing lookups,' in Proc. ACMSIGCOMM, vol. 27, pp. 25–36.
[46] Mark Baker, Rajkumar Buyya and Domenico Laforenza, The Grid: International Efforts in Global Computing
[47] N.P. Smart, \An identity based authenticated key agreement protocol based on the weil pairing," Election. Lett., Vol.38, No.13, pp.630-632, 2002
[48] Napster - http://www.napster.com/
[49] NIST, "Special Publication 800-57: Recommendation for Key Management. Part 1: General Guideline", Draft Jan.2003.
[50] Open Science Grid project. http://www.opensciencegrid.org.
[51] R. Alfieri, R. Cecchini, V. Ciaschini, L. dell'Agnello, A., Frohner, A. Gianoli, K. Lorentey, and F. Spataro. VOMS, an authorization system for virtual organizations. In Proceedings of European Access Grids Conference, pages 33–40, 2003.
[52] R. Buyya (editor), Grid Computing Info Centre, http://www.GridComputing.com/ accessed on 26-02-2010.
[53] R. Buyya (editor), High Performance Cluster Computing, Vol. 1 and 2, Prentice Hall - PTR, NJ, USA, 1999.
[54] R. Buyya, D. Abramson, and J. Giddy, A Case for Economy Grid Architecture for Service-Oriented Grid Computing, Proceedings of the International Parallel and Distributed Processing Symposium: 10th IEEE International Heterogeneous Computing Workshop (HCW 2001), April 23,

2001, San Francisco, California, USA, IEEE CS Press,USA, 2001.
[55] R. Buyya, D. Abramson, and J. Giddy, An Economy Driven Resource Management Architecture for Global Computational Power Grids, Proceedings of the 2000 International Conference on Parallel and Distributed processingTechniques and Applications (PDPTA 2000), June 26-29, 2000, Las Vegas, USA, CSREA Press, USA, 2000.
[56] R. Buyya, D. Abramson, and J. Giddy, Nimrod-G: An Architecture for a Resource Management and Scheduling System in a Global Computational Grid, The 4th International Conference on High Performance Computing in Asia- Pacific Region (HPC Asia 2000), May 2000, Beijing, China, IEEE Computer Society Press, USA.
[57] R. Buyya, S. Chapin, and D. DiNucci, Architectural Models for Resource Management in the Grid, First IEEE/ACM International Workshop on Grid Computing (GRID 2000), Springer Verlag LNCS Series, Germany, Dec. 17, 2000, Bangalore, India.
[58] R. Buyya, The Virtual Laboratory Project: Molecular Modeling for Drug Design on Grid, IEEE Distributed Systems Online, Vol. 2, No. 5, 2001. http://www.buyya.com/vlab/
[59] R. Das, J. Hanson, J. Kephart, and G. Tesauro, Agent-Human Interactions in the ContinuousDouble Auction, Proceedings of the International Joint Conferences on Artificial Intelligence (IJCAI), August 4-10, 2001, Seattle,Washington, USA.
[60] R. Metcalfe and D. Boggs, Ethernet: Distributed Packet Switching for Local Computer Networks, Proceedings of the ACM National Computer Conference Vol. 19, No. 5, July 1976.
[61] R. Smith and R. Davis, The Contract Net Protocol: High Level Communication and Control in a Distributed Problem Solver, IEEE Transactions on Computers, Vol. C-29, No. 12, pp. 1104-1113, Dec. 1980, IEEE CS Press, USA.
[62] R. Wolski, N. Spring, and J. Hayes, "The network weather service: A distributed resource performance forecasting service for metacomputing," Journal of Future Generation Computing Systems, vol. 15, pp. 757-768, 1999.
[63] Rama sangireddy, Natsuhiko Futamura, Srinivas Aluru and Arun K.Somani (Aug 2005).'Scalable, Memory Efficient, High-Speed IP Lookup Algorithms', IEEE/ACM Transactions on Networking, Vol 13, NO.4,.
[64] S. A. Vanstone, "Next generation security for wireless: elliptic curve cryptography", Computers and Security, Vol 22, No 5, Aug. 2003.
[65] S. Harris, The Tao of IETF - A Novice's Guide to the Internet Engineering Task Force, August 2001, http://www.ietf.cnri.reston.va.us/rfc/rfc3160.txt
[66] S. Nilsson and G. Karlsson (Jun. 1999) 'IP address lookup using LC-Tries,' IEEE J. Sel. Areas Commun., vol. 17, no. 6, pp. 1083–1092,.
[67] S. Smallen, W. Cirne, J. Frey, F. Berman, R. Wolski, M. Su, C. Kesselman, S. Young, and M.Ellisman, Combining Workstations and Supercomputers to Support Grid Applications: The Parallel Tomography Experience, Proceedings of the 9th Heterogeneous Computing Workshop, May 2000.
[68] S. Xian-He and W. Ming, "GHS: A performance prediction and task scheduling system for Grid computing," IEEE International Parallel and Distributed Processing Symposium (IPDPS 2003), 2003.
[69] S.A. Cook (1974): An observation on time-storage tradeoff. J. Comp. Syst. Scis. 9,308–316.
[70] SETI@Home – http://setiathome.ssl.berkeley.edu/
[71] Steve Chapin, Mark Clement, and Quinn Snell, A Grid Resource Management Architecture, Strawman 1, Grid ForumScheduling Working Group, November 1999.
[72] T. J. Norman, A. Preece, S. Chalmers, N. R. Jennigs, M. Luck,V. D. Dang, T. D. Nguyen, V. Deora, J. Shao, W. A.Gray, and N. J. Fiddian. Agent-based formation of virtual organizations.Knowledge-Based Systems, 17:103–111, 2004.
[73] The Enabling Grids for E-sciencE project. http://www.euegee.org.
[74] The Standard Performance Evaluation Corporation (SPEC), http://www.specbench.org/.
[75] United Devices, The History of Distributed Computing, http://www.ud.com/company/dc/history.htm, October 9, 2001.
[76] V. N. Padmanabhan and L. Subramanian, "An investigation of geographic mapping techniques for internet hosts," Proceedings of ACM SIGCOMM 2001- Applications, Technologies, Architectures, and Protocols for ComputersCommunications-, vol. 31, pp. 173-185, 2001.
[77] V. Srinivasan and G. Varghese (Jun. 1998) 'Fast address lookups using controlled prefix expansion,' in Proc. ACMSIGMETRICS, pp. 1–11.
[78] W. T. Sullivan, III, D. Werthimer, S. Bowyer, J. Cobb, D. Gedye, and D. Anderson, A new major SETI project based on Project Serendip data and 100,000 personal computers, Proceedings of the Fifth International Conference on Bioastronomy, 1997. http://setiathome.ssl.berkeley.edu/
[79] W. Vickrey, Counter-speculation, auctions, and competitive sealed tenders, Journal of Finance, Vol. 16, No. 1, pp. 9-37, March 1961.
[80] Y. Amir, B. Awerbuch., A. Barak A., S. Borgstrom, and A. Keren, An Opportunity Cost Approach for Job Assignment in a Scalable Computing Cluster, IEEE Transactions on Parallel and Distributed Systems, Vol. 11, No. 7, pp. 760-768, IEEE CS Press, USA, July 2000.
[81] Y. Chawathe, S. Ratnasamy, L. Breslau, N. Lanham, and S.Shenker, "Making Gnutella-like P2P Systems Scalable," Proceedings of ACM SIGCOMM 2003: Conference on Computer Communications, vol. 33, pp. 407-418, 2003.
[82] Y. Chu, S. G. Rao, and H. Zhang, "A case for end system multicast," Proceedings of ACM SIGMETRICS 2000, vol.28, pp. 1-12, 2000.
[83] Y. Liu, Z. Zhuang, L. Xiao, and L. M. Ni, "A distributed approach to solving overlay mismatching problem," Proceedings of 24th International Conference on Distributed Computing Systems, vol. 24, pp. 132-139, 2004.
[84] Z. Xu, C. Tang, and Z. Zhang, "Building topology-aware overlays using global soft-state," Proceedings of 23rd IEEE International Conference on Distributed Computing Systems, pp. 500-508, 2003.

Suresh Shanmugasundaram is a PhD Research Scholar in the Department of Computer Science and Engineering at Government College of Technology (Under Anna University, Chennai), Coimbatore, India. He is currently involved in developing an Algorithm for Gridlock avoidance. Suresh graduated his Bachelor of Engineering in the discipline of Computer Science and Engineering at Sri Ramakrishna Engineering College. Then he graduated his Master of Science in Computer Networks at Middlesex University, London. His Master Degree thesis was an approach for Secured Group Key Agreement. He has a vast experience in lecturing and tutoring the modules like Object Oriented Analysis and Design, Programming in C, Computer Networks, Management Information System, Enterprise Resource Planning, Software Engineering Methodologies, Network Security and Distributed Computing both for UG and PG Engineering students. He has attended academic award board meetings and tutor briefings held at The Open University, Milton Keynes, UK. He has been in the International Technical Committee Member for IASTED, Canada since 2008. He has chaired various International Conferences. Countries visited by him include UK, Zambia, South Africa and Namibia. He is also a reviewer for the International Journal of Theoretical and Applied Electronic Commerce Research, University of Talca, Chile from 2009. His publications include 7 National and 12 International Conference Proceedings, 1 International Journal and 1 Book Chapter.