2008

Distributed & Collaborative Key Agreement Protocol for Dynamic Peer Groups

Team Members

Ramya Natarajan

Ponmozhi Sivapatham

RANIPETTAI ENGINEERING COLLEGE

Vellore 632513

TABLE OF CONTENTS

INTRODUCTION

About Distributed Collaborative Key Agreement

With the emergence of many group-oriented distributed applications such as tele / video-conferencing and multiplayer games, there is a need for security services to provide group-oriented communication privacy and data integrity. To provide this form of group communication privacy; it is important that members of the group can establish a common group key group communication. So, we are going for a distributed group key agreement and authentication protocol so that people can establish and authenticate a common group key for private communication. This type of key agreement protocols is both distributed and contributory in nature: each member of the group contributes its part to the overall group key.

We consider several distributed collaborative key agreement and authentication protocols for dynamic peer groups. There are several important characteristics which make this problem different from traditional secure group communication.

They are: 1) distributed nature in which there is no centralized key server which generates the group key; 2) collaborative nature in which the group key is *contributory* (i.e., each group member will collaboratively contribute its part to the global group key); and 3) dynamic nature in which existing members may leave the group while new members may join. Instead of performing individual re keying operations, i.e., re computing the group key after every join or leave request, we are going to re key for a batch of join operations. Queue Batch algorithm is used to dynamically perform re-keying operation after batch of joins or leaves. The group members can communicate with the group members with the help of a common group key.

Benefits of Distributed Collaborative Key Agreement

The benefits of the distributed key agreement are to provide the members of a group with secure common group key. This group key is generated collaboratively wherein each node becomes a part of the key generation.

The distributive nature of the system, avoids the usage of a centralized key server for key generation. The dynamic nature of the system allows the existing members to leave the group while new members can join, Instead of performing individual re keying operations we are going to re key for a batch of join operations. The system uses Queue-batch algorithm for re-keying. The algorithm can substantially reduce the computation and communication workload in a highly dynamic environment. The group key is used for future communication among the members of the group.

STEM ANALYSIS

EXISTING SYSTEM:

The existing system involves centralized key server in which all the systems depend on centralized key server for key generation. All the members depend on the centralized key server for key generation. Re keying, which means renewing the keys associated with the nodes of the key tree, is performed whenever there is any group membership change including any new member joining or any existing member leaving the group.

This individual re keying operation increases the computation and communication cost. More resources have been utilized by the server in case of multiple join and leave of members in the group. For every individual re keying operation which is happening for single join and leave operation all the group members depends on the centralized key server for group key generation.

LIMITATIONS

The following are the limitations of the existing system

- All the group members depend on the centralized key server for the generation of the group key
- Individual re-keying is done. When a member joins the group or when a member leaves from the group.
- Computational and Communication cost is more when re keying is done for single join and leave operation
- More resources are used for re-keying because it is done for each join or leave operations.

PROPOSED SYSTEM

The proposed system involves collaborative key agreement in which all nodes become a part of the group key. The group key is generated which is common for all the members in the group. The communication in the group is done with the help of the group key. The members can communicate with other members such as sending files to other members with the help of the group key. Instead of performing individual re keying operations, i.e., re computing the group key after every join or leave request, we are going to re key for a batch of join operations. Moreover, re keying is done after a batch of join or leave operations**.** We consider interval-based distributed re-keying algorithms, or interval-based algorithms for short, for updating the group key:

These interval-based algorithms significantly outperform the individual re-keying approach and that the Queue-batch algorithm performs the best among the three interval-based algorithms. More importantly, the Queue-batch algorithm can substantially reduce the computation and communication workload in a highly dynamic environment. The re keying operation for multiple join members in to the group is done using Queue-batch algorithm.

The protocol remains efficient even when the occurrences of join/leave events are very frequent. Here Key information does not depend on centralized key server. The key is generated by the members in the group.

ADVANTAGES OF PROPOSED SYSTEM

- Key information does not depend on centralized key server.
- All the members will not depend on the centralized key server for key generation.
- Computational and Communication cost is less because re keying is done for batch of join and leave operations
- Resources used for re keying are minimized because it is being done for batch of join/leave operations.
- The protocol remains efficient even when the occurrences of join/leave events are very frequent.
- The Queue-batch algorithm can substantially reduce the computation and communication workload in a highly dynamic environment

HARDWARE SPECIFICATION

PROCESSOR	:	PENTIUM III 866 MHz
RAM	:	256 MB DD RAM
MONITOR	:	15” COLOR
HARD DISK	:	20 GB
FLOPPY DRIVE	:	1.44 MB
CDDRIVE	:	LG 52X
KEYBOARD	:	STANDARD 102 KEYS
MOUSE	:	3 BUTTONS

SOFTWARE SPECIFICATION

OPERATING SYSTEM	:	WINDOWS XP PROFESSIONAL
ENVIRONMENT	:	VISUAL STUDIO .NET 2003
.NET FRAMEWORK	:	VERSION 1.0
LANGUAGE	:	VISUAL BASIC.NET
BACKEND	:	SQL-SERVER 2000

MODULE DESCRIPTION:

LIST OF MODULES

1. Group Key Generation within the workgroup
2. Rekeying of group key
3. Sharing the resources within the group

MODULE 1: GROUP KEY GENERATION WITHIN THE WORKGROUP

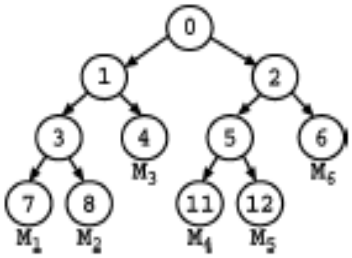
In this module we implement the Deffie-Hellman tree-based protocol to generate the group key. The tree we describe here is a binary tree where in a parent node gives rise to 2 child nodes. The private key of the leaf nodes is decided by the particular group member. The member makes a request for the public key of another child node. And once it gets it, with the knowledge of the public key of 1 child node and the private key of the other we can get the private key of its parent node using the Diffie Helman algorithm. Similarly, on proceeding higher the tree we can get the private key of the root node which is said to be the group key of the work group. In future, the entire message sent by a member to all others in the peer group is encrypted using this group key. And this module ends with the generation of the group key.

TREE-BASED GROUP DIFFIE–HELLMAN PROTOCOL

To efficiently maintain the group key in a dynamic peer group with more than two members, we use the tree-based group Diffie–Hellman (TGDH) protocol. Each member maintains a set of keys, which are arranged in a hierarchical binary tree. We assign a node ID to every tree node. For a given node, we associate a secret (or private) key Kv and a blinded (or public) key BKv. All arithmetic operations are performed in a cyclic group of prime order with the generator. Therefore, the blinded key of node can be generated by

$$BK_v = \alpha^{K_v} \bmod p.$$

Each leaf node in the tree corresponds to the individual secret and blinded keys of a group member Mi. Every member holds all the secret keys along its key path starting from its associated leaf node up to the root node. Therefore, the secret key held by the root node is shared by all the members and is regarded as the group key. The figure below illustrates a possible key tree with six members M1 to M6 . For example, member M1 and holds the keys at nodes 7, 3, 1, and 0. The secret key at node 0 is the group key of this peer group.



. Key tree used in the tree-based group Diffie–Hellman protocol.

The node ID of the root node is set to 0. Each nonleaf node consists of two child nodes whose node ID's are given by 2v+1 and 2v+2 . Based on the Diffie–Hellman protocol , the secret key of a nonleaf node can be generated by the secret key of one child node of v and the blinded key of another child node of v . Mathematically, we have

$$\begin{aligned} K_v &= (BK_{2v+1})^{K_{2v+2}} \bmod p \\ &= (BK_{2v+2})^{K_{2v+1}} \bmod p \\ &= \alpha^{K_{2v+1}K_{2v+2}} \bmod p. \end{aligned}$$

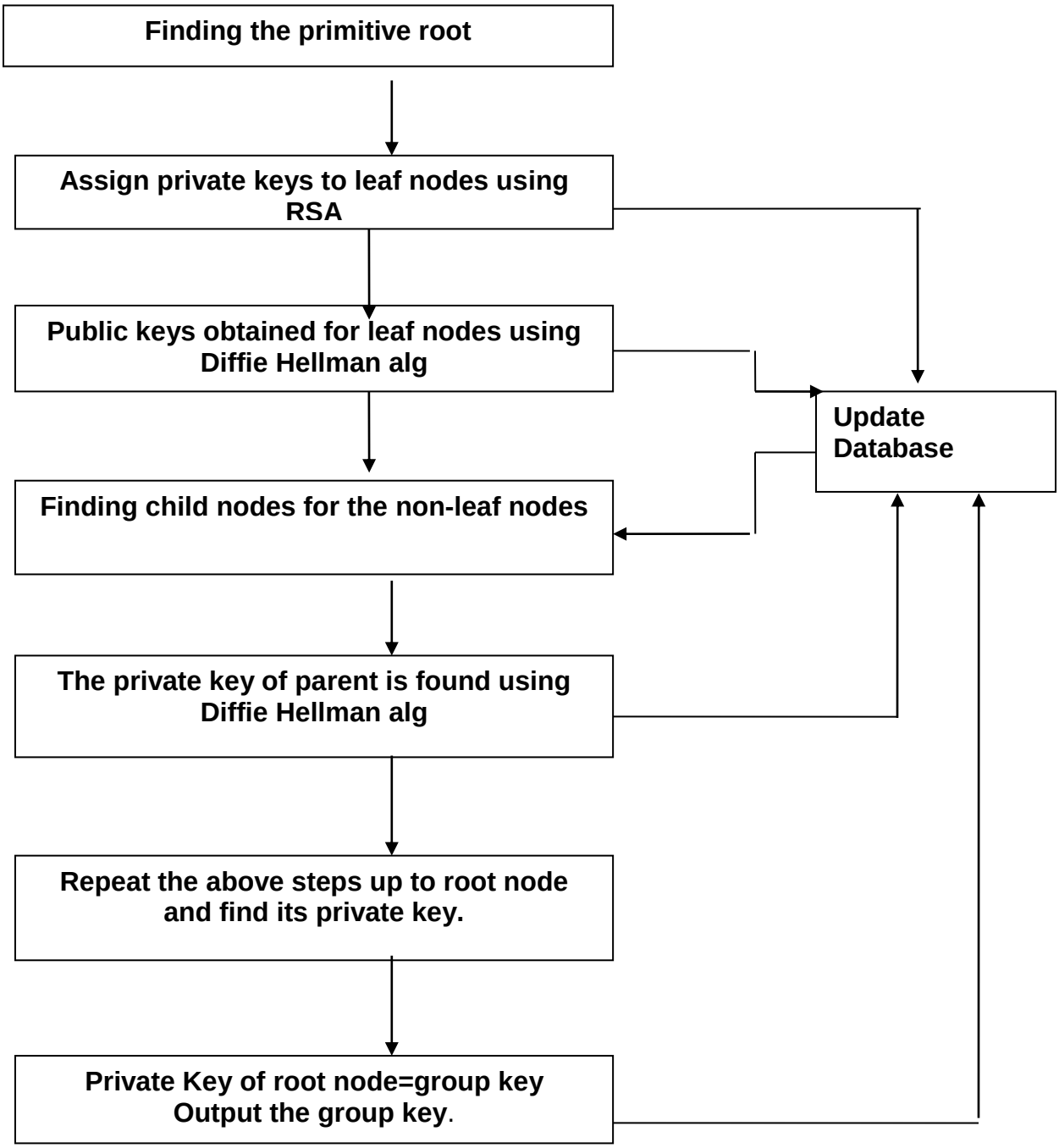
Unlike the keys at nonleaf nodes, the secret key at a leaf node is selected by its corresponding group member through a secure pseudo random number generator.Since the blinded keys are publicly known, every member can compute the keys along its key path to the root node based on its individual secret key.

To illustrate, consider the key tree in Fig. 1. Every member Mi generates its own secret key and all the secret keys along the path to the root node. For example, member M1 generates the secret key K7 and it can request the blinded key BK8 from M2, BK4 from

M3, and BK2 from either M4 , M5,or M6.Given M1's secret key K7 and the blinded key BK8, M1 can generate the secret key K3 according to the above given formula. Given the blinded key BK4 and the newly generated secret key K3 , M1 can generate the secret key K1 based on given formula. Given the secret key K1 and the blinded key BK2 , M1 can generate the secret key K0 at the root. From that point on, any communication in the group can be encrypted based on the secret key (or group key) K0.

To provide both backward confidentiality (i.e., joined member cannot access previous communication data) and forward confidentiality (i.e., left members cannot access future com-munication data), rekeying, which means renewing the keys associated with the nodes of the key tree, is performed when-ever there is any group membership change including any new member joining or any existing member leaving the group.

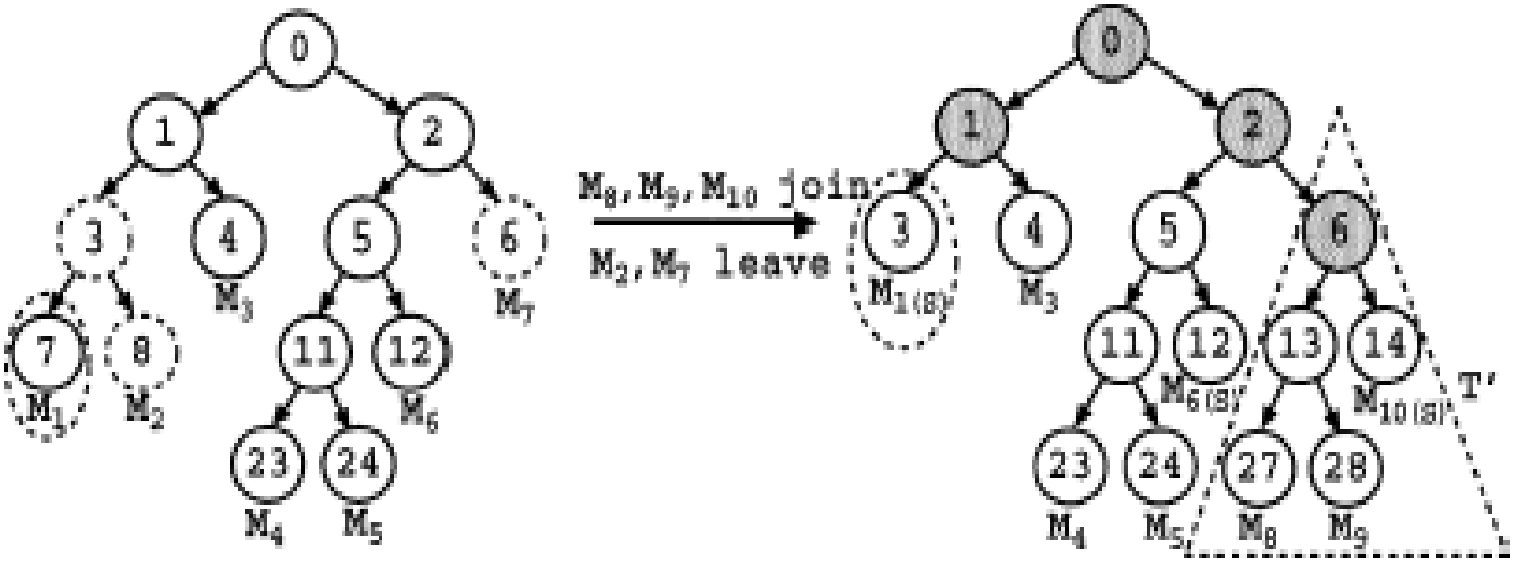
FLOW CHART FOR MODULE 1



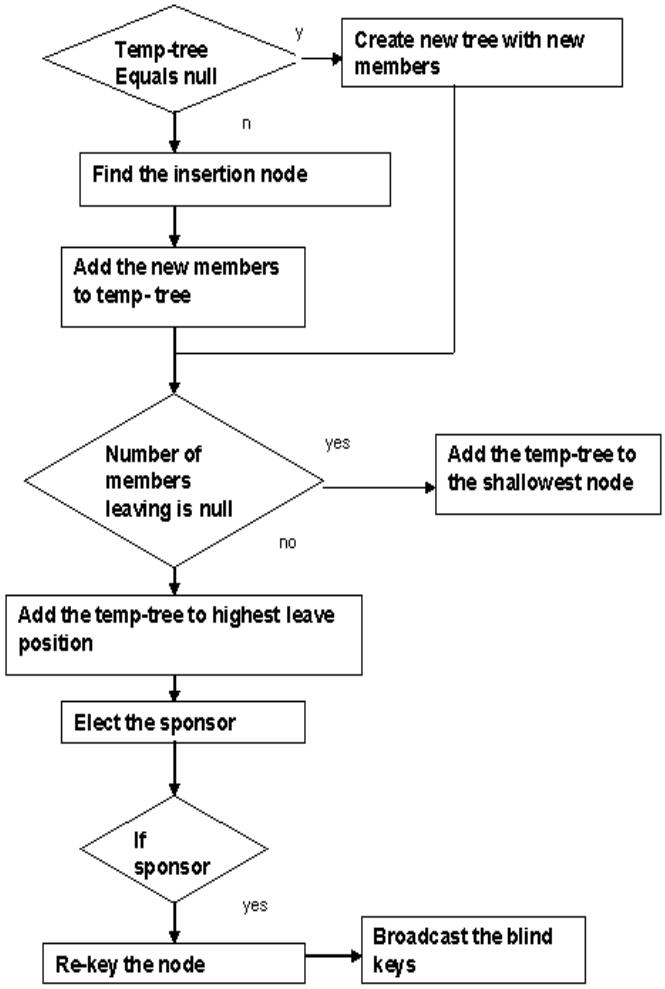
MODULE 2: REKEYING OF GROUP KEY

Queue batch algorithm, an interval-based algorithm is used for re-keying at equal intervals. Queue-batch algorithm performs the best among the interval-based algorithms. The algorithm reduces the latency and the workload created due to re-keying operation that is performed at the beginning of the re-keying intervals. In Queue batch algorithm, as and when members join, they are stored as in a temporary tree and at the beginning of a re-keying interval this tree is attached to the tree with existing members. It is attached to the highest departed position, so that the height of the tree does not increase much.

The Queue- batch algorithm is illustrated in Figure, where members M8, M9, M10 wish to join the communication group, while M2 and M7 wish to leave. Then in the Queue-sub tree phase, the three new members M8, M9, M10 will form a tree. In the Queue-merge phase, the tree is added at the highest departed position, which is at node 6. Now group key is computed for the new group structure and the computed group key is broadcasted to all the members.


0
1
2
3
4
5
6
7
8
11
12
23
24
M1
M2
M3
M4
M5
M6
M7
M8, M9, M10 join
M2, M7 leave
M1(8)'
M6(8)'
M10(8)'
13
14
27
28
M8
M9
T'

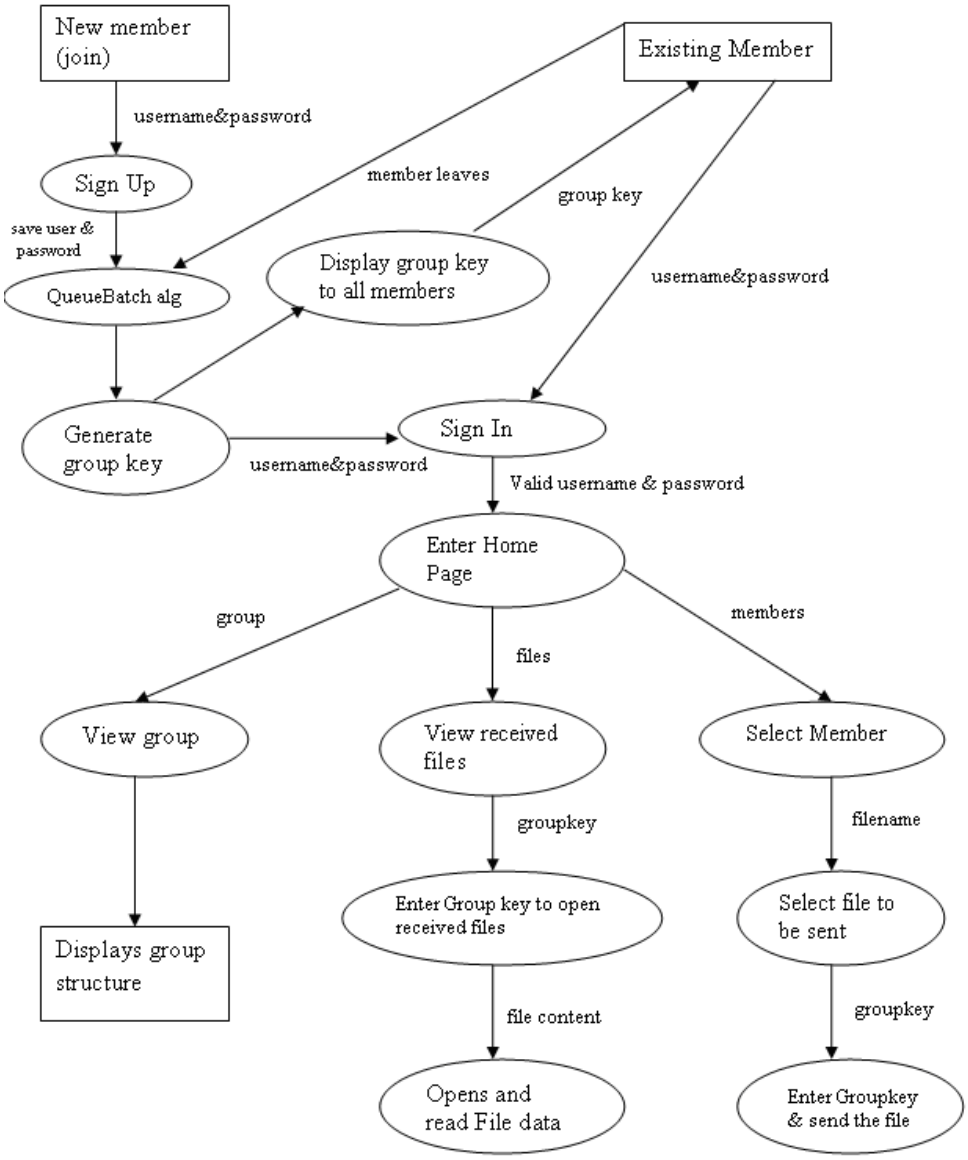
FLOW CHART FOR MODULE 2



MODULE 3: SHARING THE RESOURCES WITHIN THE GROUP

The new group key is been generated after the batch of join and leave using the Queue Batch algorithm in the 2nd module. From now onwards this new group key is used for encryption for all data sharing among the members of the peer group. In this module we would be able to show all the communication and data sharing among all the members present in our work group.

3 Data Flow Diagram



SERVER IMPLEMENTATION:

GLOBAL.VS

```
Module global

    Public alpha As Integer = 2
    Public prime As Integer = 17
    Public servername As String = Environment.MachineName.ToString
    Public serverport As Integer = 8000
    Public clientport As Integer = 9000
    Public clientname As String = ""
    Public status As Boolean = False
    Public members() As String
    Public pbkey() As String
    Public count As Integer
    Public rprime() = {2, 3, 5, 7, 11, 13, 17, 19, 23, 29}
    Public rndprime As New Random(10)
    Public ralpha As New Random(5)

End Module
```

ASSEMBLY INFO:

```
Imports System
Imports System.Reflection
Imports System.Runtime.InteropServices

' General Information about an assembly is controlled through the following
' set of attributes. Change these attribute values to modify the information
' associated with an assembly.

' Review the values of the assembly attributes

<Assembly: AssemblyTitle("")>
<Assembly: AssemblyDescription("")>
<Assembly: AssemblyCompany("")>
<Assembly: AssemblyProduct("")>
<Assembly: AssemblyCopyright("")>
<Assembly: AssemblyTrademark("")>
<Assembly: CLSCompliant(True)>

'The following GUID is for the ID of the typelib if this project is exposed to COM
<Assembly: Guid("56FD77A5-0D16-499E-A011-38A1F38503E2")>
```

```
' Version information for an assembly consists of the following four values:
'
'      Major Version
'      Minor Version
'      Build Number
'      Revision
'
' You can specify all the values or you can default the Build and Revision Numbers
' by using the '*' as shown below:

<Assembly: AssemblyVersion("1.0.*")>


Imports System
Imports System.Runtime.Remoting.Channels
Imports System.Net
Imports System.Runtime.Remoting.Messaging
Imports System.Collections

Class IpInjectorSink
   Inherits BaseChannelObjectWithProperties
   Implements IServerChannelSink
   Private _nextSink As IServerChannelSink

   Public Sub New(ByVal nextSink As IServerChannelSink)
      _nextSink = nextSink
   End Sub

   Public Overrides ReadOnly Property Properties() As IDictionary Implements
IChannelSinkBase.Properties
      Get
         Return MyBase.Properties
      End Get
   End Property

   Public Sub AsyncProcessResponse(ByVal sinkStack As
System.Runtime.Remoting.Channels.IServerResponseChannelSinkStack, ByVal state As
Object, ByVal msg As System.Runtime.Remoting.Messaging.IMessage, ByVal headers
As System.Runtime.Remoting.Channels.ITransportHeaders, ByVal stream As
System.IO.Stream) Implements
System.Runtime.Remoting.Channels.IServerChannelSink.AsyncProcessResponse

      Try
         Dim IPAddr As IPAddress = headers(CommonTransportKeys.IPAddress)
         CallContext.SetData("ClientIP", IPAddr)
      Catch IPAddrEx As Exception
```

```
            'do nothing
        End Try

        'forward to stack for further processing
        sinkStack.AsyncProcessResponse(msg, headers, stream)
    End Sub

    Public Function GetResponseStream(ByVal sinkStack As
System.Runtime.Remoting.Channels.IServerResponseChannelSinkStack, ByVal state As
Object, ByVal msg As System.Runtime.Remoting.Messaging.IMessage, ByVal headers
As System.Runtime.Remoting.Channels.ITransportHeaders) As System.IO.Stream
Implements
System.Runtime.Remoting.Channels.IServerChannelSink.GetResponseStream

        Return Nothing
    End Function

    Public ReadOnly Property NextChannelSink() As
System.Runtime.Remoting.Channels.IServerChannelSink Implements
System.Runtime.Remoting.Channels.IServerChannelSink.NextChannelSink
        Get
            Return _nextSink
        End Get
    End Property

    Public Function ProcessMessage(ByVal sinkStack As
System.Runtime.Remoting.Channels.IServerChannelSinkStack, ByVal requestMsg As
System.Runtime.Remoting.Messaging.IMessage, ByVal requestHeaders As
System.Runtime.Remoting.Channels.ITransportHeaders, ByVal requestStream As
System.IO.Stream, ByRef responseMsg As
System.Runtime.Remoting.Messaging.IMessage, ByRef responseHeaders As
System.Runtime.Remoting.Channels.ITransportHeaders, ByRef responseStream As
System.IO.Stream) As System.Runtime.Remoting.Channels.ServerProcessing
Implements System.Runtime.Remoting.Channels.IServerChannelSink.ProcessMessage

        Try
            Dim IPAddr As IPAddress = requestHeaders(CommonTransportKeys.IPAddress)
            CallContext.SetData("ClientIP", IPAddr)
        Catch IPAddrEx As Exception
            'do nothing
        End Try
        sinkStack.Push(Me, Nothing)

        Dim srvProc As ServerProcessing = _nextSink.ProcessMessage(sinkStack,
requestMsg, requestHeaders, requestStream, responseMsg, responseHeaders,
responseStream)
```

```
        If srvProc = ServerProcessing.Complete Then
            'TODO - implement post processing
        End If
        Return srvProc
    End Function

End Class

Class IpInjectorSinkProvider
    Implements IServerChannelSinkProvider

    Private _nextProvider As IServerChannelSinkProvider

    Public Sub New()
```

KEY:

```
Public Class key
    Public Function publickey(ByVal alpha As Double, ByVal prime As Double, ByVal
privatekey As Double) As Double
        'Formula PBkey= (Alpha (pow) privaltekey)mod prime
        Dim result1 As Double
        result1 = Math.Pow(alpha, privatekey)
        result1 = result1 Mod prime
        Return result1
        'Return (Math.Pow(alpha, privatekey)) Mod prime
    End Function
    Public Function sharedkey(ByVal publickey As Double, ByVal privatekey As Double,
ByVal prime As Double) As Double
        'Formula sharedkey= (publickey (pow) privaltekey)mod prime
        Dim result As Double
        result = Math.Pow(publickey, privatekey)
        result = result Mod prime
        Return result
    End Function
End Class

Imports System.Data.SqlClient
'Imports System.Data.OleDb
Module Module1
    Public rnd As New Random(40)
    Public members() As String
    Public pbkey() As String
    Public count As Integer
    Public dbpath As String = System.Windows.Forms.Application.StartupPath
```

```
    'Public connection As New
OleDbConnection("Provider=Microsoft.Jet.OLEDB.4.0;Data Source= " & dbpath &
"\Keyagreement.mdb;")
    'Public connection As New OleDbConnection("Provider=SQLOLEDB;Data
Source=.;Initial Catalog=keyagrement;UserId=sa;Password=;")

    Public connection As New
SqlConnection("Server=.;Database=keyagrement;UID=sa;Pwd=;")
    Public Function readdb() As String()
        Dim str(50) As String
        Dim cmd As New SqlCommand
        Dim cmd1 As New SqlCommand
        If connection.State = ConnectionState.Closed Then
            connection.Open()
        End If

        cmd.Connection = connection
        cmd.CommandType = CommandType.Text
        cmd.CommandText = "select * from userdb"

        Dim obj, obj1 As SqlDataReader
        Dim i As Integer = 0
        Dim count As Integer = 0
        obj = cmd.ExecuteReader()
        While obj.Read
            count = count + 1
        End While
        If connection.State = ConnectionState.Open Then
            connection.Close()
        End If
        If connection.State = ConnectionState.Closed Then
            connection.Open()
        End If

        cmd1.Connection = connection
        cmd1.CommandType = CommandType.Text
        cmd1.CommandText = "select * from userdb"
        str(0) = Convert.ToString(count)
        obj1 = cmd1.ExecuteReader()
        While obj1.Read
            i = i + 1
            str(i) = obj1(0)
            str(i + count) = obj1(1)
            str(i + (count * 2)) = obj1(2)
            str(i + (count * 3)) = obj1(3)
        End While
```

```
    If connection.State = ConnectionState.Open Then
        connection.Close()
    End If
    Return str
End Function


Public Function readdb1() As String()
    Dim str(100) As String
    Dim cmd As New SqlCommand
    Dim cmd1 As New SqlCommand
    If connection.State = ConnectionState.Closed Then
        connection.Open()
    End If

    cmd.Connection = connection
    cmd.CommandType = CommandType.Text
    cmd.CommandText = "select filename from filedb"

    Dim obj, obj1 As SqlDataReader
    Dim i As Integer = 0
    Dim count As Integer = 0
    obj = cmd.ExecuteReader()
    While obj.Read
        count = count + 1
    End While
    If connection.State = ConnectionState.Open Then
        connection.Close()
    End If
    If connection.State = ConnectionState.Closed Then
        connection.Open()
    End If

    cmd1.Connection = connection
    cmd1.CommandType = CommandType.Text
    cmd1.CommandText = "select filename from filedb"
    str(0) = Convert.ToString(count)
    obj1 = cmd1.ExecuteReader()
    While obj1.Read
        i = i + 1
        str(i) = obj1(0)
    End While
    If connection.State = ConnectionState.Open Then
        connection.Close()
    End If
    Return str
```

```
    End Function
    Public Function getvalues(ByVal q As String) As String
        Dim str As String
        Dim cmd As New SqlCommand
        Dim cmd1 As New SqlCommand
        If connection.State = ConnectionState.Closed Then
            connection.Open()
        End If

        cmd.Connection = connection
        cmd.CommandType = CommandType.Text
        cmd.CommandText = q
        Dim obj As SqlDataReader
        Dim i As Integer = 0
        Dim count As Integer = 0
        obj = cmd.ExecuteReader()
        While obj.Read
            str = obj(0)
        End While
        If connection.State = ConnectionState.Open Then
            connection.Close()
        End If


        Return str
    End Function

End Module
```

SERVER IMPLEMENTATION:

```
Public Class Server
    Inherits System.Windows.Forms.Form
    Dim objServer As ServerInterface.ServerInterface
#Region " Windows Form Designer generated code "

    Public Sub New()
        MyBase.New()

        'This call is required by the Windows Form Designer.
        InitializeComponent()

        'Add any initialization after the InitializeComponent() call

    End Sub
```

```
    'Form overrides dispose to clean up the component list.
    Protected Overloads Overrides Sub Dispose(ByVal disposing As Boolean)
        If disposing Then
            If Not (components Is Nothing) Then
                components.Dispose()
            End If
        End If
        MyBase.Dispose(disposing)
    End Sub

    'Required by the Windows Form Designer
    Private components As System.ComponentModel.IContainer


        '
        'Label1
        '
        Me.Label1.Font = New System.Drawing.Font("Microsoft Sans Serif", 9.0!,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, CType(0,
Byte))
        Me.Label1.Location = New System.Drawing.Point(112, 24)
        Me.Label1.Name = "Label1"
        Me.Label1.Size = New System.Drawing.Size(64, 16)
        Me.Label1.TabIndex = 21
        Me.Label1.Text = "Members"
        '
        'Button3
        '
        Me.Button3.Enabled = False
        Me.Button3.Location = New System.Drawing.Point(120, 40)
        Me.Button3.Name = "Button3"
        Me.Button3.Size = New System.Drawing.Size(80, 32)
        Me.Button3.TabIndex = 28
        Me.Button3.Text = "STOP"
        '
        'Button1
        '
        Me.Button1.Location = New System.Drawing.Point(32, 40)
        Me.Button1.Name = "Button1"
        Me.Button1.Size = New System.Drawing.Size(80, 32)
        Me.Button1.TabIndex = 27
        Me.Button1.Text = "START"
        '
        'GroupBox3
        '
```

```
Me.GroupBox3.BackColor = System.Drawing.Color.RosyBrown
Me.GroupBox3.Controls.Add(Me.ComboBox1)
Me.GroupBox3.Controls.Add(Me.DataGrid1)
Me.GroupBox3.Location = New System.Drawing.Point(8, 248)
Me.GroupBox3.Name = "GroupBox3"
Me.GroupBox3.Size = New System.Drawing.Size(320, 232)
Me.GroupBox3.TabIndex = 26
Me.GroupBox3.TabStop = False
'
'ComboBox1
'
Me.ComboBox1.Items.AddRange(New Object() {"Nodedb", "Userdb", "filedb"})
Me.ComboBox1.Location = New System.Drawing.Point(16, 16)
Me.ComboBox1.Name = "ComboBox1"
Me.ComboBox1.Size = New System.Drawing.Size(192, 21)
Me.ComboBox1.TabIndex = 19
'
'DataGrid1
'
Me.DataGrid1.DataMember = ""
Me.DataGrid1.HeaderForeColor = System.Drawing.SystemColors.ControlText
Me.DataGrid1.Location = New System.Drawing.Point(16, 40)
Me.DataGrid1.Name = "DataGrid1"
Me.DataGrid1.Size = New System.Drawing.Size(296, 176)
Me.DataGrid1.TabIndex = 18
'
'Button5
'
Me.Button5.Location = New System.Drawing.Point(208, 40)
Me.Button5.Name = "Button5"
Me.Button5.Size = New System.Drawing.Size(80, 32)
Me.Button5.TabIndex = 25
Me.Button5.Text = "SETTINGS"
'
'GroupBox4
'
Me.GroupBox4.Controls.Add(Me.PictureBox1)
Me.GroupBox4.Location = New System.Drawing.Point(8, 0)
Me.GroupBox4.Name = "GroupBox4"
Me.GroupBox4.Size = New System.Drawing.Size(600, 504)
Me.GroupBox4.TabIndex = 1
Me.GroupBox4.TabStop = False
Me.GroupBox4.Text = "Tree Structure"
'
'PictureBox1
'
```

```
        Me.PictureBox1.BackColor = System.Drawing.Color.White
        Me.PictureBox1.Location = New System.Drawing.Point(8, 16)
        Me.PictureBox1.Name = "PictureBox1"
        Me.PictureBox1.Size = New System.Drawing.Size(584, 480)
        Me.PictureBox1.TabIndex = 0
        Me.PictureBox1.TabStop = False
        '
        'Server
        '
        Me.AutoScaleBaseSize = New System.Drawing.Size(5, 13)
        Me.BackColor = System.Drawing.Color.RosyBrown
        Me.ClientSize = New System.Drawing.Size(960, 517)
        Me.Controls.Add(Me.GroupBox4)
        Me.Controls.Add(Me.GroupBox1)
        Me.Name = "Server"
        Me.Text = "Server"
        Me.GroupBox1.ResumeLayout(False)
        Me.GroupBox2.ResumeLayout(False)
        Me.GroupBox3.ResumeLayout(False)
        CType(Me.DataGrid1, System.ComponentModel.ISupportInitialize).EndInit()
        Me.GroupBox4.ResumeLayout(False)
        Me.ResumeLayout(False)

    End Sub

#End Region
    Private Sub Server_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Load

    End Sub
    <System.Runtime.Remoting.Messaging.OneWay()> _
        Public Sub ReceiveLibCall(ByVal message As String)
        ' MsgBox("Client is called")
    End Sub
    <System.Runtime.Remoting.Messaging.OneWay()> _
    Public Sub ReceiveNewClient(ByVal host As String)
        ClientList.Items.Add(host)
    End Sub
    <System.Runtime.Remoting.Messaging.OneWay()> _
   Public Sub Receivegetmembers()
        Dim i As Integer
        ' For i = 0 To ListBox2.Items.Count - 1
        'Next
    End Sub
    <System.Runtime.Remoting.Messaging.OneWay()> _
   Public Sub Receivefile(ByVal fname As String, ByVal str1 As String)
```

```
        Try
            'MsgBox(fname.ToString)
            'MsgBox(str1.ToString)
            ServerModule.insert1("insert into Filedb values('" & fname.ToString & "','" &
str1.ToString & "')")
            'MsgBox("New File is upload")
        Catch ex As Exception
        End Try
    End Sub
    <System.Runtime.Remoting.Messaging.OneWay()> _
    Public Sub ReceiveNewUser(ByVal uname As String, ByVal hname As String, ByVal
prkey As String)
        Dim obj As New Keygeneration.key
        Dim bkey As String = (obj.publickey(alpha, prime, prkey)).ToString()
        ListBox2.Items.Add(uname)
        ListBox1.Items.Add(bkey)
        ClientList.Items.Add(hname)
        Dim str As String
        '    count = ListBox1.Items.Count
        ' Dim i As Integer
        'For i = 0 To count - 1
        Dim userid As Integer = ServerModule.autonumber("uid", "userdb")
        If userid = 0 Then
            userid = 1
        End If
        str = "insert into userdb(uid,username,publickey,hostname) values(" & userid & ",'"
& uname & "'," & bkey & ",'" & hname & "')"
        ServerModule.insert1(str)
        str = "insert into userdb1(uid,pkey) values(" & userid & "," & prkey & ")"
        ServerModule.insert1(str)

        '   Servertoclient.Init()
        Try
            'client.refreshrequest()
        Catch e1 As Exception
            'MsgBox(e1.ToString())
            Label4.Text = e1.ToString
        End Try
    End Sub
    <System.Runtime.Remoting.Messaging.OneWay()> _
    Public Sub deleteuser(ByVal id As String, ByVal mname As String)
        Dim nodeid As Integer = ServerModule.getvalue("select nodeid from userdb where
uid=" & id & " ")
        Dim lid As Integer = ServerModule.getvalue("select lid from nodedb where nodeid
= " & nodeid & "")
```

```
        Dim rid As Integer = ServerModule.getvalue("select rid from nodedb where nodeid
= " & nodeid & "")
        Dim newnodeid As Integer = ServerModule.getvalue("select nodeid from nodedb
where lid = " & nodeid & " or rid=" & nodeid & "")
        ServerModule.insert1("delete  from userdb where uid=" & id & "")
        ServerModule.insert1("delete  from nodedb where nodeid=" & nodeid & "")
        If lid = id Then
            '\\ delete
            ServerModule.insert1("update nodedb set lid=" & rid & " where lid = " & nodeid
& " or rid = " & nodeid & " ")
            ServerModule.insert1("update nodedb set rid=" & rid & " where lid = " & nodeid
& " or rid = " & nodeid & " ")
            ServerModule.insert1("update userdb set nodeid=" & newnodeid &
",level1=level1-1 where id=" & rid & " ")
        ElseIf rid = id Then
            ServerModule.insert1("update nodedb set lid=" & lid & " where lid = " & nodeid
& " or rid = " & nodeid & " ")
            ServerModule.insert1("update nodedb set rid=" & lid & " where lid = " & nodeid
& " or rid = " & nodeid & " ")
            ServerModule.insert1("update userdb set nodeid=" & newnodeid &
",level1=level1-1 where uid=" & lid & " ")
        End If
        referesh()
    End Sub
    <System.Runtime.Remoting.Messaging.OneWay()> _
        Public Sub Receivejoinuser(ByVal tuid As String, ByVal tuname As String, ByVal
thname As String, ByVal fusername As String, ByVal fhname As String, ByVal prkey As
String)
        Dim obj As New Keygeneration.key
        Dim fbkey As String = obj.publickey(alpha, prime, prkey)
        Dim skey As String = obj.sharedkey(fbkey, prkey, prime)
        clientname = thname
        Servertoclient.Init()
        Dim n As Integer = 1
        'n = client.newjoinrequest(fusername, fbkey, skey)
        If n = 1 Then
            Try

                Dim nodebkey As Double = obj.publickey(alpha, prime, skey)
                '  Servertoclient.Init()
                'client.refreshrequest()

                Dim nodeid As Integer = ServerModule.autonumber("nodeid", "nodedb")
                If nodeid = 0 Then
                    nodeid = 51
                End If
```

```
            Dim oldnodeid As Integer = ServerModule.getvalue("select nodeid from userdb
where uid=" & tuid & "")

            Dim newuserid As Integer = ServerModule.autonumber("uid", "userdb")

            Dim lid As Integer = ServerModule.selectid(tuname)
            Dim rid As Integer = newuserid
            Dim level As Integer = ServerModule.getvalue("select level1 from userdb
where uid=" & lid & "")

            ServerModule.insert1("update nodedb set lid = " & nodeid & " where lid=" &
tuid & "")
            ServerModule.insert1("update nodedb set rid = " & nodeid & " where rid=" &
tuid & "")

            Dim str As String = "insert into nodedb values(" & nodeid & "," & skey & ","
& nodebkey & "," & lid & "," & rid & "," & level & ")"
            Dim re As Integer = ServerModule.insert1(str)

            Dim str1 As String = "update userdb set nodeid=" & nodeid & ", level1=" &
level + 1 & " where uid=" & lid & " "
            ServerModule.insert1(str1)

            str = "insert into userdb(uid,username,publickey,hostname,nodeid,level1)
values(" & newuserid & ",'" & fusername & "'," & fbkey & ",'" & fhname & "'," &
nodeid & "," & level + 1 & ")"
            ServerModule.insert1(str)
            str = "insert into userdb1(uid,pkey) values(" & newuserid & "," & prkey & ")"
            ServerModule.insert1(str)
            rekying()
            ListBox2.Items.Add(fusername)
            ClientList.Items.Add(fhname)
            ListBox1.Items.Add(fbkey)
            loaddbtotree()
          Catch ex As Exception
            'MsgBox(ex.ToString)
            Label4.Text = ex.ToString
          End Try
        End If
    End Sub
    Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs)
        '   Servertoclient.Init()
        '  MsgBox(Servertoclient.client.Test(10, 10))
    End Sub
```

```
    Private Sub Button2_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs)
        '  Servertoclient.Init()
        '  Servertoclient.client.Test(10, 10)
    End Sub
    Private Sub Button1_Click_2(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button1.Click
        Try
            ServerModule.Serverstart()
            AddHandler ServerModule.server.LibCalled, AddressOf ReceiveLibCall
            AddHandler ServerModule.server.EventnewClient, AddressOf ReceiveNewClient
            AddHandler ServerModule.server.EventnewUser, AddressOf ReceiveNewUser
            AddHandler ServerModule.server.Eventgetmembers, AddressOf
Receivegetmembers
            AddHandler ServerModule.server.Eventjoinuser, AddressOf Receivejoinuser
            AddHandler ServerModule.server.Eventsavefile, AddressOf Receivefile
            AddHandler ServerModule.server.Eventdeleteuser, AddressOf deleteuser

            ServerModule.deleteuser("userdb")
            ServerModule.deleteuser("nodedb")
            ServerModule.deleteuser("userdb1")

            ClientList.Items.Clear()
            ListBox1.Items.Clear()
            ListBox2.Items.Clear()
            MsgBox("SERVER IS STARTED")
            Button1.Enabled = False
            Button3.Enabled = True
            GroupBox2.Enabled = True
            GroupBox3.Enabled = True
            GroupBox4.Enabled = True
        Catch ex As Exception
            Label4.Text = ex.ToString()
        End Try
    End Sub
    Private Sub Button3_Click_1(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button3.Click
        If MsgBox(" Server is stoped") = MsgBoxResult.OK Then
            GroupBox2.Enabled = False
            GroupBox3.Enabled = False
            GroupBox4.Enabled = False
            Button1.Enabled = True
            Button3.Enabled = False
        End If
    End Sub
```

```
    Private Sub ComboBox1_SelectedIndexChanged_1(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles ComboBox1.SelectedIndexChanged
        Dim obj As New dlclass
        Dim dt As DataTable = ServerModule.load(ComboBox1.SelectedItem)
        DataGrid1.DataSource = dt
    End Sub
    Private Sub Button5_Click_1(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button5.Click
        Dim frm As New Settings
        frm.Show()
    End Sub
    Private Function referesh()
        Dim str() As String = readdb()
        Dim i As Integer
        ClientList.Items.Clear()
        ListBox2.Items.Clear()
        ListBox1.Items.Clear()
        For i = 1 To str(0)
            ClientList.Items.Add(str(i + (str(0) * 3)))
            ListBox1.Items.Add(str(i + (str(0) * 2)))
            ListBox2.Items.Add(str(i + (str(0))))
        Next
    End Function
    Private Function rekying()
        Dim obj As New Keygeneration.key
        Dim maxlevel = (autonumber("level1", "Userdb")) - 1
        Dim i As Integer
        Dim j As Integer
        Dim lid As Integer
        Dim rid As Integer
        Dim nodeid As Integer
        Dim shkey1 As String
        Dim pbkey As String
        Dim prkey As String
        If maxlevel > 1 Then
            For i = maxlevel To 2 Step -1
                Dim value() As Integer = countfield((i - 1), "nodeDB", "level")
                For j = 1 To value(0)
                    nodeid = value(j)
                    lid = value(value(0) + j)
                    rid = value((value(0) * 2) + j)
                    If nodeid > 50 Then
                        If (lid > 50) Then
                            pbkey = ServerModule.getvalue("select publickey from nodedb where
nodeid=" & rid & "")
```

```
                    prkey = ServerModule.getvalue("select groupkey from nodedb where
nodeid=" & lid & "")
                Else
                    pbkey = ServerModule.getvalue("select publickey from userdb where
uid=" & rid & "")
                    prkey = ServerModule.getvalue("select pkey from userdb1 where uid="
& lid & "")
                End If
                'If (rid > 50) Then  Else  End If
                shkey1 = obj.sharedkey(pbkey, prkey, prime)
                ServerModule.insert1("update nodedb set groupkey = " & shkey1 & "
where nodeid = " & nodeid & " ")
                Dim npublickey As String = obj.publickey(alpha, prime, shkey1)
                ServerModule.insert1("update nodedb set publickey = " & npublickey & "
where nodeid = " & nodeid & " ")
            End If
        Next
    Next
  End If
End Function
Public Function draw(ByVal tree() As Integer, ByVal maxlevel As Integer)
  Dim i As Integer
  Dim j As Integer
  Dim counter = 1
  Dim number As Integer = 1
  Dim y As Integer = 50
  Dim w As Integer = PictureBox1.Width
  Dim x As Integer = w / 2
  Dim z As Integer = x
  Dim myPen = New Pen(Color.Green, 2)
  Dim mypen1 = New Pen(Color.Black, 1)
  Dim bmp01 As New Bitmap(PictureBox1.Width, PictureBox1.Height)
  PictureBox1.Image = bmp01
  Dim bmp As New Bitmap(PictureBox1.Image)
  Dim graphics As Graphics = graphics.FromImage(bmp)
  Dim fnt As System.Drawing.Font
  Dim brush As System.Drawing.Brush
  fnt = New System.Drawing.Font("Arial", FontStyle.Bold)
  'Dim x2 = PictureBox1.Width / 2

  Dim fontFmly As New FontFamily("Tahoma")
  Dim green28 As New Font(fontFmly, 10)
  Dim red14Italic As New Font(fontFmly, 10, FontStyle.Italic)

  For i = 1 To maxlevel
    For j = 1 To number
```

```
            If tree(counter) <> 0 Then
               graphics.DrawEllipse(myPen, x, y, 30, 20)
               graphics.DrawString(tree(counter).ToString, green28, New
SolidBrush(Color.Red), x + 5, y + 5)
               If tree(counter) > 50 Then
                  Dim v As Integer
                  v = x - (z / 2)
                  graphics.DrawLine(mypen1, x + 15, y + 20, v + 5, (y + 60))
                  v = x + (z / 2)
                  graphics.DrawLine(mypen1, x + 15, y + 20, v + 5, (y + 60))
               End If
               '  graphics.DrawString(tree(counter).ToString, fnt, brush, x, y)
            End If
            counter = counter + 1
            x = x + (z * 2)

         Next
         number = number * 2
         y = y + 60
         x = w / (number * 2)
         z = x
      Next

      'graphics.DrawEllipse(myPen, 50, 50, 50, 100)
      'graphics.DrawLine(mypen1, 10, 10, 20, 100)
      PictureBox1.Image = bmp
      graphics.Dispose()
   End Function
   Public Function loaddbtotree() As Integer()
      Dim tree(100) As Integer
      Dim maxlevel = (autonumber("level1", "userdb")) - 1
      Dim rootnodeid = ServerModule.getvalue("select nodeid from nodedb where
level=1")
      Dim i, j As Integer
      Dim number As Integer = 2
      Dim id As Integer
      Dim gid As Integer
      Dim gnumber As Integer
      If maxlevel > 1 Then
         tree(1) = rootnodeid
         For i = 1 To maxlevel - 1
            gnumber = number / 2
            For j = 1 To number

               If (j Mod 2) <> 0 Then
                  gid = tree((gnumber))
```

```
                If gid > 50 Then
                    id = ServerModule.getvalue("select lid from nodedb where nodeid = "
& gid & " ")
                Else
                    id = 0
                End If
                tree(number + (j - 1)) = id
            Else
                gid = tree(gnumber)
                If gid > 50 Then
                    id = ServerModule.getvalue("select rid from nodedb where nodeid = "
& gid & " ")
                Else
                    id = 0
                End If
                tree(number + (j - 1)) = id
                gnumber = gnumber + 1
            End If
        Next
        number = number * 2
    Next
  End If

  draw(tree, maxlevel)
  'Dim myPen = New Pen(Color.Green, 2)
  'Dim mypen1 = New Pen(Color.Gray, 2)

  'Dim bmp01 As New Bitmap(PictureBox1.Width, PictureBox1.Height)
  'PictureBox1.Image = bmp01
  'Dim bmp As New Bitmap(PictureBox1.Image)
  'Dim graphics As Graphics = graphics.FromImage(bmp)
  'graphics.DrawEllipse(myPen, 50, 50, 50, 100)
  'graphics.DrawLine(mypen1, 10, 10, 20, 100)
  'PictureBox1.Image = bmp
  'graphics.Dispose()
End Function
Public Function interval()
  Dim thrd As System.Threading.Thread
  thrd.Start()
  thrd.Sleep(10)
  prime = rprime(rndprime.Next())
  alpha = ralpha.Next()
End Function
```

```
    Private Sub GroupBox1_Enter(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles GroupBox1.Enter

    End Sub
End Class

Imports System.Runtime.Remoting
Imports System.Runtime.Remoting.Channels.Tcp
Imports System.Runtime.Remoting.Channels
Module Servertoclient
    Public client As ClientInterface.Clientinterface
    Private Channel2 As System.Runtime.Remoting.Channels.tcp.TcpChannel 'New
System.Runtime.Remoting.Channels.tcp.TcpChannel(0)
    Private serverProv As BinaryServerFormatterSinkProvider
    Private clientProv As BinaryClientFormatterSinkProvider
    Private props As IDictionary = New Hashtable

    'Public clienthostname = "sunbeam1"
    ' Public clienthostname = clientname

    Private Sub SetChannel()
        serverProv = New BinaryServerFormatterSinkProvider
        clientProv = New BinaryClientFormatterSinkProvider
        serverProv.TypeFilterLevel =
System.Runtime.Serialization.Formatters.TypeFilterLevel.Full
        props("port") = 0
        'props("name") = privServiceName
        Channel2 = New TcpChannel(props, clientProv, serverProv)
    End Sub
    Public Sub Init()
        SetChannel()
        'ChannelServices.RegisterChannel(Channel2)
        Try
            client = CType(Activator.GetObject(GetType(ClientInterface.Clientinterface),
"tcp://" & clientname & ":9000/Client"), ClientInterface.Clientinterface)
            If client Is Nothing Then
                'MsgBox("client Not connected")
            End If
        Catch ex As Exception
            ' MsgBox(ex.Message)
        End Try
        'RemotingConfiguration.Configure("SimpleClient.exe.config")
        'server = New RemoteLib
    End Sub
    Public Sub dispose()
        ChannelServices.UnregisterChannel(Channel2)
```

```
        client = Nothing
        Channel2 = Nothing
        serverProv = Nothing
        clientProv = Nothing
    End Sub
    ' Public m As Integer = 0

End Module

Public Class Settings
    Inherits System.Windows.Forms.Form

#Region " Windows Form Designer generated code "

    Public Sub New()
        MyBase.New()

        'This call is required by the Windows Form Designer.
        InitializeComponent()

        'Add any initialization after the InitializeComponent() call

    End Sub

    'Form overrides dispose to clean up the component list.
    Protected Overloads Overrides Sub Dispose(ByVal disposing As Boolean)
        If disposing Then
            If Not (components Is Nothing) Then
                components.Dispose()
            End If
        End If
        MyBase.Dispose(disposing)
    End Sub

    'Required by the Windows Form Designer
    Private components As System.ComponentModel.IContainer

    'NOTE: The following procedure is required by the Windows Form Designer
    'It can be modified using the Windows Form Designer.
    'Do not modify it using the code editor.
    Friend WithEvents Label1 As System.Windows.Forms.Label
    Friend WithEvents Label2 As System.Windows.Forms.Label
    Friend WithEvents Label3 As System.Windows.Forms.Label
    Friend WithEvents TextBox1 As System.Windows.Forms.TextBox
    Friend WithEvents TextBox2 As System.Windows.Forms.TextBox
    Friend WithEvents TextBox3 As System.Windows.Forms.TextBox
```

```
Friend WithEvents Button1 As System.Windows.Forms.Button
<System.Diagnostics.DebuggerStepThrough()> Private Sub InitializeComponent()
    Me.Label1 = New System.Windows.Forms.Label
    Me.Label2 = New System.Windows.Forms.Label
    Me.Label3 = New System.Windows.Forms.Label
    Me.TextBox1 = New System.Windows.Forms.TextBox
    Me.TextBox2 = New System.Windows.Forms.TextBox
    Me.TextBox3 = New System.Windows.Forms.TextBox
    Me.Button1 = New System.Windows.Forms.Button
    Me.SuspendLayout()
    '
    'Label1
    '
    Me.Label1.Location = New System.Drawing.Point(24, 48)
    Me.Label1.Name = "Label1"
    Me.Label1.TabIndex = 0
    Me.Label1.Text = "ServerName"
    '
    'Label2
    '
    Me.Label2.Location = New System.Drawing.Point(24, 96)
    Me.Label2.Name = "Label2"
    Me.Label2.TabIndex = 1
    Me.Label2.Text = "Portno"
    '
    'Label3
    '
    Me.Label3.Location = New System.Drawing.Point(24, 144)
    Me.Label3.Name = "Label3"
    Me.Label3.TabIndex = 2
    Me.Label3.Text = "Client Portno"
    '
    'TextBox1
    '
    Me.TextBox1.Enabled = False
    Me.TextBox1.Location = New System.Drawing.Point(160, 48)
    Me.TextBox1.Name = "TextBox1"
    Me.TextBox1.TabIndex = 3
    Me.TextBox1.Text = ""
    '
    'TextBox2
    '
    Me.TextBox2.Enabled = False
    Me.TextBox2.Location = New System.Drawing.Point(160, 96)
    Me.TextBox2.Name = "TextBox2"
    Me.TextBox2.TabIndex = 4
```

```
        Me.TextBox2.Text = "8000"
        '
        'TextBox3
        '
        Me.TextBox3.Enabled = False
        Me.TextBox3.Location = New System.Drawing.Point(160, 144)
        Me.TextBox3.Name = "TextBox3"
        Me.TextBox3.TabIndex = 5
        Me.TextBox3.Text = "9000"
        '
        'Button1
        '
        Me.Button1.Location = New System.Drawing.Point(96, 192)
        Me.Button1.Name = "Button1"
        Me.Button1.Size = New System.Drawing.Size(96, 24)
        Me.Button1.TabIndex = 6
        Me.Button1.Text = "OK"
        '
        'Settings
        '
        Me.AutoScaleBaseSize = New System.Drawing.Size(5, 13)
        Me.BackColor = System.Drawing.Color.RosyBrown
        Me.ClientSize = New System.Drawing.Size(312, 221)
        Me.Controls.Add(Me.Button1)
        Me.Controls.Add(Me.TextBox3)
        Me.Controls.Add(Me.TextBox2)
        Me.Controls.Add(Me.TextBox1)
        Me.Controls.Add(Me.Label3)
        Me.Controls.Add(Me.Label2)
        Me.Controls.Add(Me.Label1)
        Me.Name = "Settings"
        Me.Text = "Settings"
        Me.ResumeLayout(False)

    End Sub

#End Region
    Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button1.Click
        'servername = TextBox1.Text
        'serverport = TextBox2.Text
        'clientport = TextBox3.Text
        Me.Close()
    End Sub
    Private Sub Settings_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Load
```

```
        TextBox1.Text = servername
        TextBox2.Text = serverport
        TextBox3.Text = clientport
    End Sub
End Class
```

CLIENT IMPLEMENTATION:

```
Public Class NewUserfrm
    Inherits System.Windows.Forms.Form
#Region " Windows Form Designer generated code "

    Public Sub New()
        MyBase.New()

        'This call is required by the Windows Form Designer.
        InitializeComponent()

        'Add any initialization after the InitializeComponent() call

    End Sub

    'Form overrides dispose to clean up the component list.
    Protected Overloads Overrides Sub Dispose(ByVal disposing As Boolean)
        If disposing Then
            If Not (components Is Nothing) Then
                components.Dispose()
            End If
        End If
        MyBase.Dispose(disposing)
    End Sub

    'Required by the Windows Form Designer
    Private components As System.ComponentModel.IContainer

    'NOTE: The following procedure is required by the Windows Form Designer
    'It can be modified using the Windows Form Designer.
    'Do not modify it using the code editor.
    Friend WithEvents Usernametxt As System.Windows.Forms.TextBox
    Friend WithEvents Passwordtxt As System.Windows.Forms.TextBox
    Friend WithEvents Keyvaluetxt As System.Windows.Forms.TextBox
    Friend WithEvents Usernamelbl As System.Windows.Forms.Label
    Friend WithEvents Passwordlbl As System.Windows.Forms.Label
    Friend WithEvents Keyvaluelbl As System.Windows.Forms.Label
```

```
Friend WithEvents GroupBox3 As System.Windows.Forms.GroupBox
Friend WithEvents GroupBox2 As System.Windows.Forms.GroupBox
Friend WithEvents Label2 As System.Windows.Forms.Label
Friend WithEvents Button3 As System.Windows.Forms.Button
Friend WithEvents Button4 As System.Windows.Forms.Button
Friend WithEvents Button6 As System.Windows.Forms.Button
Friend WithEvents Label1 As System.Windows.Forms.Label
Friend WithEvents uidtxt As System.Windows.Forms.TextBox
<System.Diagnostics.DebuggerStepThrough()> Private Sub InitializeComponent()
    Me.Usernametxt = New System.Windows.Forms.TextBox
    Me.Passwordtxt = New System.Windows.Forms.TextBox
    Me.Keyvaluetxt = New System.Windows.Forms.TextBox
    Me.Usernamelbl = New System.Windows.Forms.Label
    Me.Passwordlbl = New System.Windows.Forms.Label
    Me.Keyvaluelbl = New System.Windows.Forms.Label
    Me.GroupBox3 = New System.Windows.Forms.GroupBox
    Me.Button3 = New System.Windows.Forms.Button
    Me.Button4 = New System.Windows.Forms.Button
    Me.Button6 = New System.Windows.Forms.Button
    Me.GroupBox2 = New System.Windows.Forms.GroupBox
    Me.Label2 = New System.Windows.Forms.Label
    Me.Label1 = New System.Windows.Forms.Label
    Me.uidtxt = New System.Windows.Forms.TextBox
    Me.GroupBox3.SuspendLayout()
    Me.GroupBox2.SuspendLayout()
    Me.SuspendLayout()
    '
    'Usernametxt
    '
    Me.Usernametxt.BorderStyle = System.Windows.Forms.BorderStyle.FixedSingle
    Me.Usernametxt.Location = New System.Drawing.Point(152, 72)
    Me.Usernametxt.Name = "Usernametxt"
    Me.Usernametxt.Size = New System.Drawing.Size(144, 20)
    Me.Usernametxt.TabIndex = 0
    Me.Usernametxt.Text = ""
    '
    'Passwordtxt
    '
    Me.Passwordtxt.BorderStyle = System.Windows.Forms.BorderStyle.FixedSingle
    Me.Passwordtxt.Location = New System.Drawing.Point(152, 104)
    Me.Passwordtxt.Name = "Passwordtxt"
    Me.Passwordtxt.PasswordChar = Microsoft.VisualBasic.ChrW(42)
    Me.Passwordtxt.Size = New System.Drawing.Size(144, 20)
    Me.Passwordtxt.TabIndex = 1
    Me.Passwordtxt.Text = ""
    '
```

```
'Keyvaluetxt
'
Me.Keyvaluetxt.BorderStyle = System.Windows.Forms.BorderStyle.FixedSingle
Me.Keyvaluetxt.Location = New System.Drawing.Point(152, 136)
Me.Keyvaluetxt.Name = "Keyvaluetxt"
Me.Keyvaluetxt.Size = New System.Drawing.Size(144, 20)
Me.Keyvaluetxt.TabIndex = 2
Me.Keyvaluetxt.Text = ""
'
'Usernamelbl
'
Me.Usernamelbl.BackColor = System.Drawing.Color.RosyBrown
Me.Usernamelbl.Location = New System.Drawing.Point(32, 72)
Me.Usernamelbl.Name = "Usernamelbl"
Me.Usernamelbl.TabIndex = 3
Me.Usernamelbl.Text = "USER NAME"
'
'Passwordlbl
'
Me.Passwordlbl.BackColor = System.Drawing.Color.RosyBrown
Me.Passwordlbl.Location = New System.Drawing.Point(32, 104)
Me.Passwordlbl.Name = "Passwordlbl"
Me.Passwordlbl.TabIndex = 4
Me.Passwordlbl.Text = "PASSWORD"
'
'Keyvaluelbl
'
Me.Keyvaluelbl.BackColor = System.Drawing.Color.RosyBrown
Me.Keyvaluelbl.Location = New System.Drawing.Point(32, 136)
Me.Keyvaluelbl.Name = "Keyvaluelbl"
Me.Keyvaluelbl.Size = New System.Drawing.Size(80, 23)
Me.Keyvaluelbl.TabIndex = 5
Me.Keyvaluelbl.Text = "PRIVATEKEY"
'
'GroupBox3
'
Me.GroupBox3.BackColor = System.Drawing.Color.RosyBrown
Me.GroupBox3.Controls.Add(Me.Button3)
Me.GroupBox3.Controls.Add(Me.Button4)
Me.GroupBox3.Controls.Add(Me.Button6)
Me.GroupBox3.Location = New System.Drawing.Point(0, 240)
Me.GroupBox3.Name = "GroupBox3"
Me.GroupBox3.Size = New System.Drawing.Size(328, 80)
Me.GroupBox3.TabIndex = 11
Me.GroupBox3.TabStop = False
'
```

```
'Button3
'
Me.Button3.FlatStyle = System.Windows.Forms.FlatStyle.Flat
Me.Button3.Location = New System.Drawing.Point(208, 16)
Me.Button3.Name = "Button3"
Me.Button3.Size = New System.Drawing.Size(104, 40)
Me.Button3.TabIndex = 17
Me.Button3.Text = "EXIT"
'
'Button4
'
Me.Button4.FlatStyle = System.Windows.Forms.FlatStyle.Flat
Me.Button4.Location = New System.Drawing.Point(120, 16)
Me.Button4.Name = "Button4"
Me.Button4.Size = New System.Drawing.Size(104, 40)
Me.Button4.TabIndex = 16
Me.Button4.Text = "DELETE"
'
'Button6
'
Me.Button6.FlatStyle = System.Windows.Forms.FlatStyle.Flat
Me.Button6.Location = New System.Drawing.Point(24, 16)
Me.Button6.Name = "Button6"
Me.Button6.Size = New System.Drawing.Size(104, 40)
Me.Button6.TabIndex = 15
Me.Button6.Text = "SAVE"
'
'GroupBox2
'
Me.GroupBox2.BackColor = System.Drawing.Color.RosyBrown
Me.GroupBox2.Controls.Add(Me.uidtxt)
Me.GroupBox2.Controls.Add(Me.Label1)
Me.GroupBox2.Controls.Add(Me.Usernamelbl)
Me.GroupBox2.Controls.Add(Me.Usernametxt)
Me.GroupBox2.Controls.Add(Me.Passwordlbl)
Me.GroupBox2.Controls.Add(Me.Passwordtxt)
Me.GroupBox2.Controls.Add(Me.Keyvaluelbl)
Me.GroupBox2.Controls.Add(Me.Keyvaluetxt)
Me.GroupBox2.Location = New System.Drawing.Point(0, 56)
Me.GroupBox2.Name = "GroupBox2"
Me.GroupBox2.Size = New System.Drawing.Size(328, 176)
Me.GroupBox2.TabIndex = 10
Me.GroupBox2.TabStop = False
'
'Label2
'
```

```
        Me.Label2.Font = New System.Drawing.Font("Microsoft Sans Serif", 12.0!,
System.Drawing.FontStyle.Bold)
        Me.Label2.Location = New System.Drawing.Point(16, 16)
        Me.Label2.Name = "Label2"
        Me.Label2.Size = New System.Drawing.Size(272, 32)
        Me.Label2.TabIndex = 15
        Me.Label2.Text = "REGISTRATION"
        Me.Label2.TextAlign = System.Drawing.ContentAlignment.MiddleCenter
        '
        'Label1
        '
        Me.Label1.BackColor = System.Drawing.Color.RosyBrown
        Me.Label1.Location = New System.Drawing.Point(32, 40)
        Me.Label1.Name = "Label1"
        Me.Label1.TabIndex = 6
        Me.Label1.Text = "USERID"
        '
        'uidtxt
        '
        Me.uidtxt.BorderStyle = System.Windows.Forms.BorderStyle.FixedSingle
        Me.uidtxt.Location = New System.Drawing.Point(152, 40)
        Me.uidtxt.Name = "uidtxt"
        Me.uidtxt.Size = New System.Drawing.Size(144, 20)
        Me.uidtxt.TabIndex = 7
        Me.uidtxt.Text = ""
        '
        'NewUserfrm
        '
        Me.AutoScaleBaseSize = New System.Drawing.Size(5, 13)
        Me.BackColor = System.Drawing.Color.RosyBrown
        Me.ClientSize = New System.Drawing.Size(336, 333)
        Me.Controls.Add(Me.Label2)
        Me.Controls.Add(Me.GroupBox3)
        Me.Controls.Add(Me.GroupBox2)
        Me.Name = "NewUserfrm"
        Me.StartPosition = System.Windows.Forms.FormStartPosition.CenterScreen
        Me.Text = "NewUserfrm"
        Me.GroupBox3.ResumeLayout(False)
        Me.GroupBox2.ResumeLayout(False)
        Me.ResumeLayout(False)

    End Sub

#End Region
    Dim Blobj As New BLuser
```

```
    Dim str As String
    Private Sub Savebtn_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs)
        Dim user As Integer
        Try
            'MsgBox(server.Test(10, 10))
            ClientModule.Init()
            ' ClientModule.server.newUser(Usernametxt.Text, Passwordtxt.Text,
"1111111111")
            ' ClientModule.dispose()
            user = Blobj.Inseruser(Usernametxt.Text, Passwordtxt.Text)
            If user = 1 Then
                ' MsgBox("saved")
            Else
                ' MsgBox("not saved")
            End If
        Catch ex As Exception
            MsgBox(ex.Message)
        End Try

    End Sub

    Private Sub Deletebtn_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs)

    End Sub

    Private Sub NewUserfrm_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Load
        ' Module1.m = 1
        'MessageBox.Show(Module1.m)

        '  Button2.Enabled = False


    End Sub

    Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs)
        'Button2.Enabled = True
        " Private Key user input
        'If Keyvaluetxt.Text = "" Then
        '    MsgBox("Please Enter The Privat Key in numbers")
        'Else
        '    ' TextBox1.Text = Math.Pow(alpha, Keyvaluetxt.Text) Mod
Convert.ToDouble(prime)
```

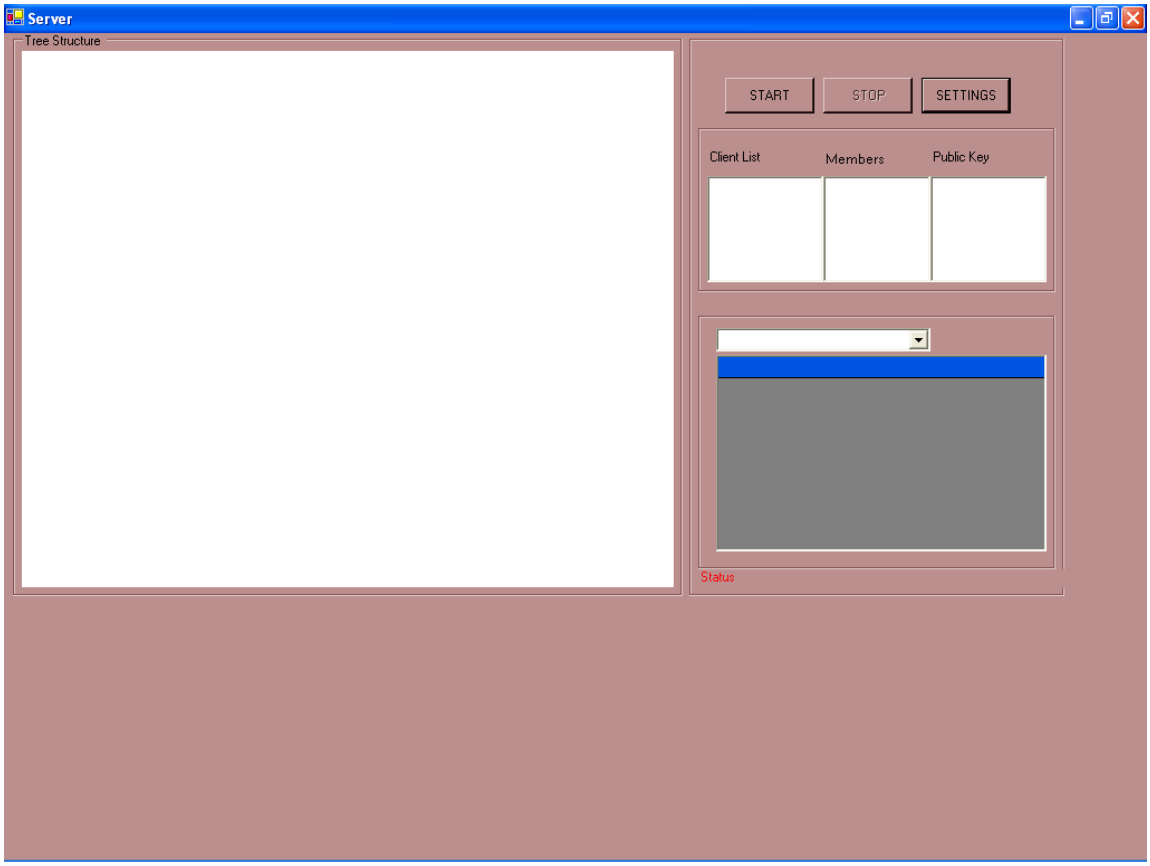
```
        'End If
    End Sub

    Private Sub Button2_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs)

    End Sub

    Private Sub GroupBox2_Enter(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles GroupBox2.Enter

    End Sub

    Private Sub Button6_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button6.Click
        Dim uid As String = uidtxt.Text
        Dim uname As String = Usernametxt.Text
        Dim pwd As String = Passwordtxt.Text
        Dim pkey As Double = Keyvaluetxt.Text
        str = " insert into userdb values('" & uid & "','" & uname & "','" & pwd & "'," &
pkey & ")"
        ' Dim value As Integer = global.insertuser(str)
        ' If value = 1 Then
        '  'MsgBox(" Saved Successfully")
        ' Else
        '     MsgBox("Save Failed")
        '  End If

    End Sub

    Private Sub Button3_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button3.Click
        Me.Close()
    End Sub

    Private Sub Button4_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button4.Click
        Dim uid As String = uidtxt.Text
        str = " delete from userdb where userid= '" & uid & "'"
        ' Dim value As Integer = global.insertuser(str)
        ' If value = 1 Then
        'MsgBox("Deleted Successfully")
        ' Else
        '     MsgBox("Deletion Failed")
        '  End If
```

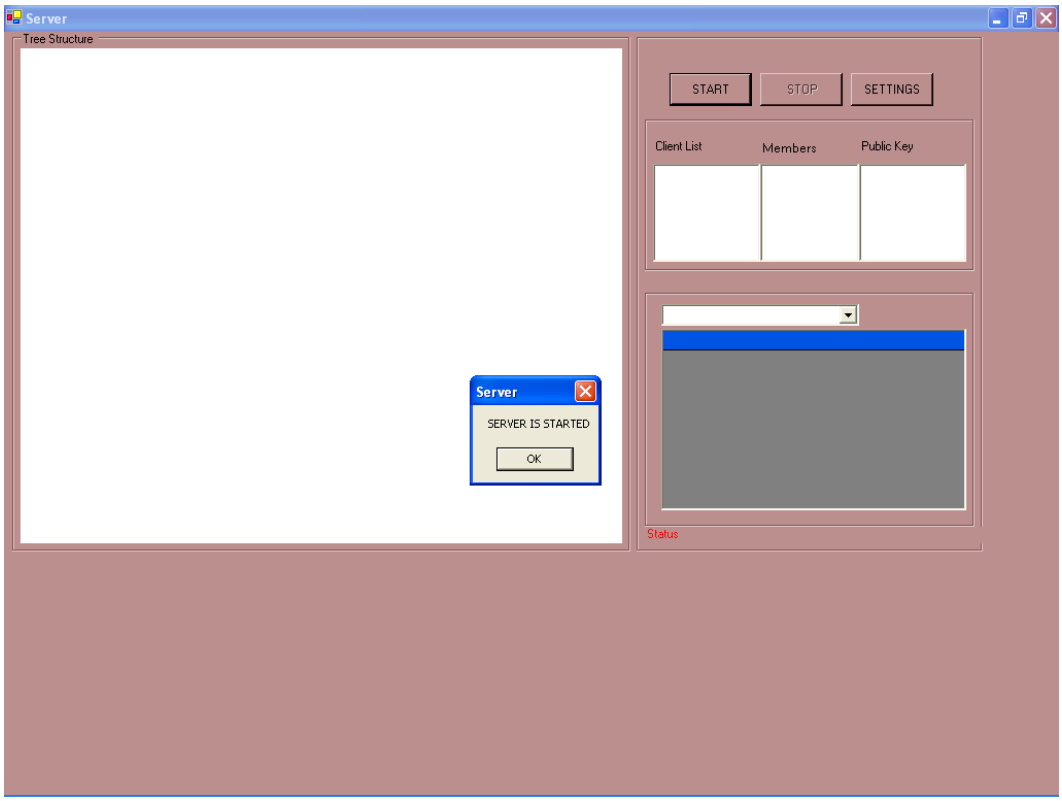
```
    End Sub
End Class

Imports System
Imports System.Runtime.Remoting.Channels
Imports System.Net
Imports System.Runtime.Remoting.Messaging
Imports System.Collections

Class IpInjectorSink
    Inherits BaseChannelObjectWithProperties
    Implements IServerChannelSink
    Private _nextSink As IServerChannelSink

    Public Sub New(ByVal nextSink As IServerChannelSink)
        _nextSink = nextSink
    End Sub

    Public Overrides ReadOnly Property Properties() As IDictionary Implements
IChannelSinkBase.Properties
        Get
            Return MyBase.Properties
        End Get
    End Property

    Public Sub AsyncProcessResponse(ByVal sinkStack As
System.Runtime.Remoting.Channels.IServerResponseChannelSinkStack, ByVal state As
Object, ByVal msg As System.Runtime.Remoting.Messaging.IMessage, ByVal headers
As System.Runtime.Remoting.Channels.ITransportHeaders, ByVal stream As
System.IO.Stream) Implements
System.Runtime.Remoting.Channels.IServerChannelSink.AsyncProcessResponse

        Try
            Dim IPAddr As IPAddress = headers(CommonTransportKeys.IPAddress)
            CallContext.SetData("ClientIP", IPAddr)
        Catch IPAddrEx As Exception
            'do nothing
        End Try

        'forward to stack for further processing
        sinkStack.AsyncProcessResponse(msg, headers, stream)
    End Sub

    Public Function GetResponseStream(ByVal sinkStack As
System.Runtime.Remoting.Channels.IServerResponseChannelSinkStack, ByVal state As
Object, ByVal msg As System.Runtime.Remoting.Messaging.IMessage, ByVal headers
```

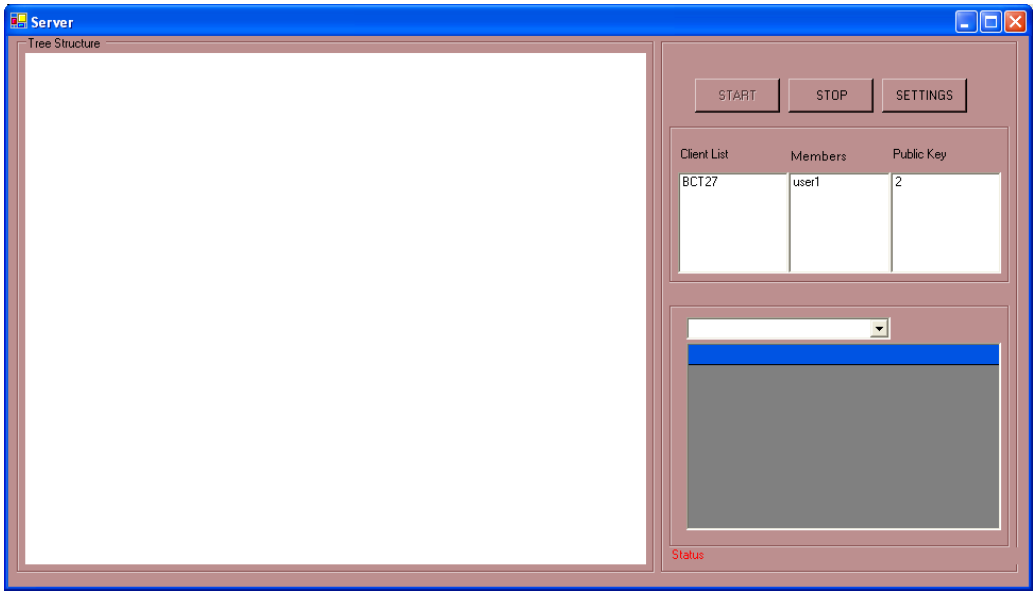
```
As System.Runtime.Remoting.Channels.ITransportHeaders) As System.IO.Stream
Implements
System.Runtime.Remoting.Channels.IServerChannelSink.GetResponseStream

        Return Nothing
    End Function

    Public ReadOnly Property NextChannelSink() As
System.Runtime.Remoting.Channels.IServerChannelSink Implements
System.Runtime.Remoting.Channels.IServerChannelSink.NextChannelSink
        Get
            Return _nextSink
        End Get
    End Property

    Public Function ProcessMessage(ByVal sinkStack As
System.Runtime.Remoting.Channels.IServerChannelSinkStack, ByVal requestMsg As
System.Runtime.Remoting.Messaging.IMessage, ByVal requestHeaders As
System.Runtime.Remoting.Channels.ITransportHeaders, ByVal requestStream As
System.IO.Stream, ByRef responseMsg As
System.Runtime.Remoting.Messaging.IMessage, ByRef responseHeaders As
System.Runtime.Remoting.Channels.ITransportHeaders, ByRef responseStream As
System.IO.Stream) As System.Runtime.Remoting.Channels.ServerProcessing
Implements System.Runtime.Remoting.Channels.IServerChannelSink.ProcessMessage

        Try
            Dim IPAddr As IPAddress = requestHeaders(CommonTransportKeys.IPAddress)
            CallContext.SetData("ClientIP", IPAddr)
        Catch IPAddrEx As Exception
            'do nothing
        End Try

        sinkStack.Push(Me, Nothing)

        Dim srvProc As ServerProcessing = _nextSink.ProcessMessage(sinkStack,
requestMsg, requestHeaders, requestStream, responseMsg, responseHeaders,
responseStream)

        If srvProc = ServerProcessing.Complete Then
            'TODO - implement post processing
        End If

        Return srvProc
    End Function

End Class
```

```
Class IpInjectorSinkProvider
   Implements IServerChannelSinkProvider

   Private _nextProvider As IServerChannelSinkProvider

   Public Sub New()

   End Sub

   Public Sub New(ByVal properties As IDictionary, ByVal ByValproviderdata As
ICollection)

   End Sub

   Public Function CreateSink(ByVal channel As
System.Runtime.Remoting.Channels.IChannelReceiver) As
System.Runtime.Remoting.Channels.IServerChannelSink Implements
System.Runtime.Remoting.Channels.IServerChannelSinkProvider.CreateSink

      Dim nextSink As IServerChannelSink = _nextProvider.CreateSink(channel)

      Return New IpInjectorSink(nextSink)

   End Function

   Public Property [Next]() As
System.Runtime.Remoting.Channels.IServerChannelSinkProvider Implements
System.Runtime.Remoting.Channels.IServerChannelSinkProvider.Next
      Get
         Return _nextProvider
      End Get
      Set(ByVal Value As
System.Runtime.Remoting.Channels.IServerChannelSinkProvider)
         _nextProvider = Value
      End Set
   End Property

   Public Sub GetChannelData(ByVal channelData As
System.Runtime.Remoting.Channels.IChannelDataStore) Implements
System.Runtime.Remoting.Channels.IServerChannelSinkProvider.GetChannelData

   End Sub

End Class
```

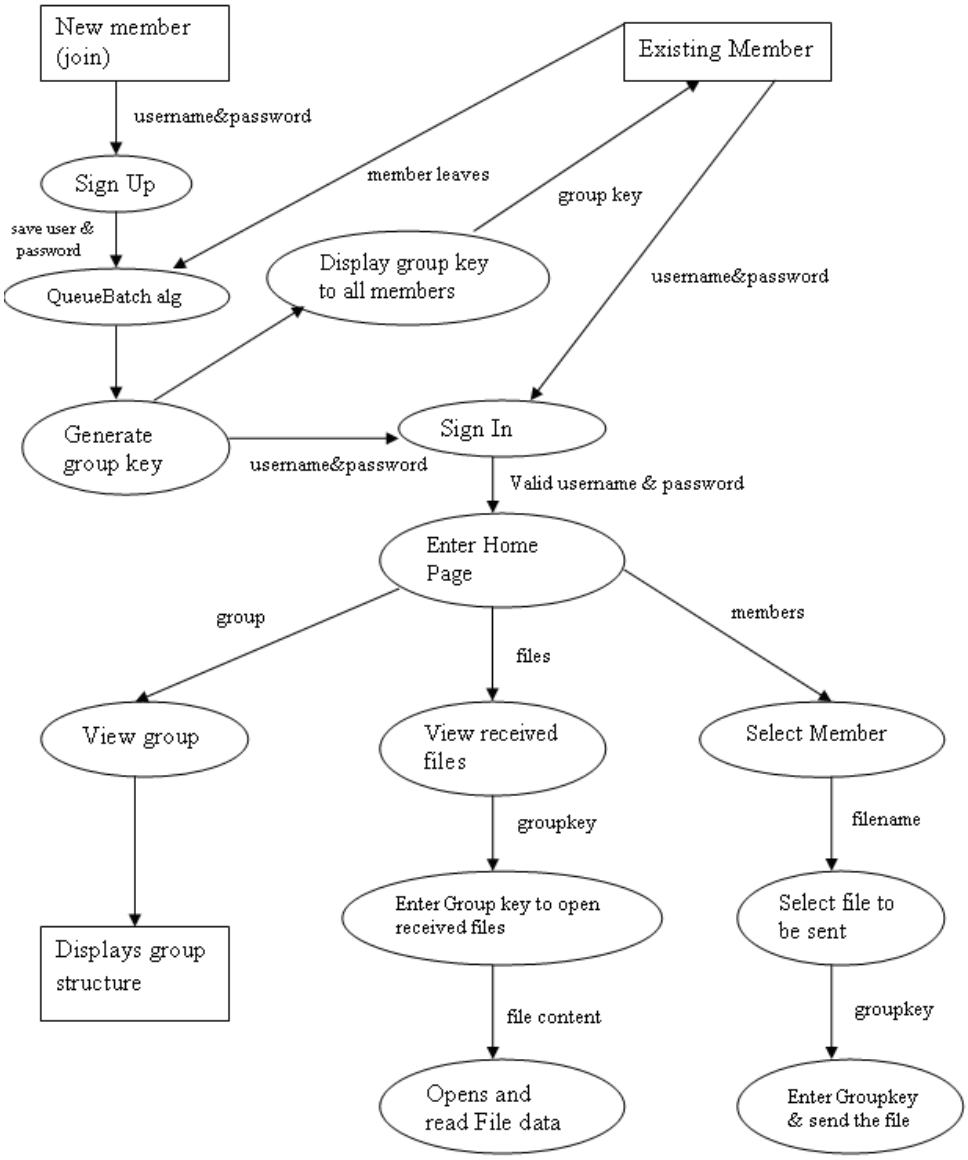
SCREEN SHOTS




Server
Tree Structure
START
STOP
SETTINGS
Client List
Members
Public Key
Server
SERVER IS STARTED
OK
Status

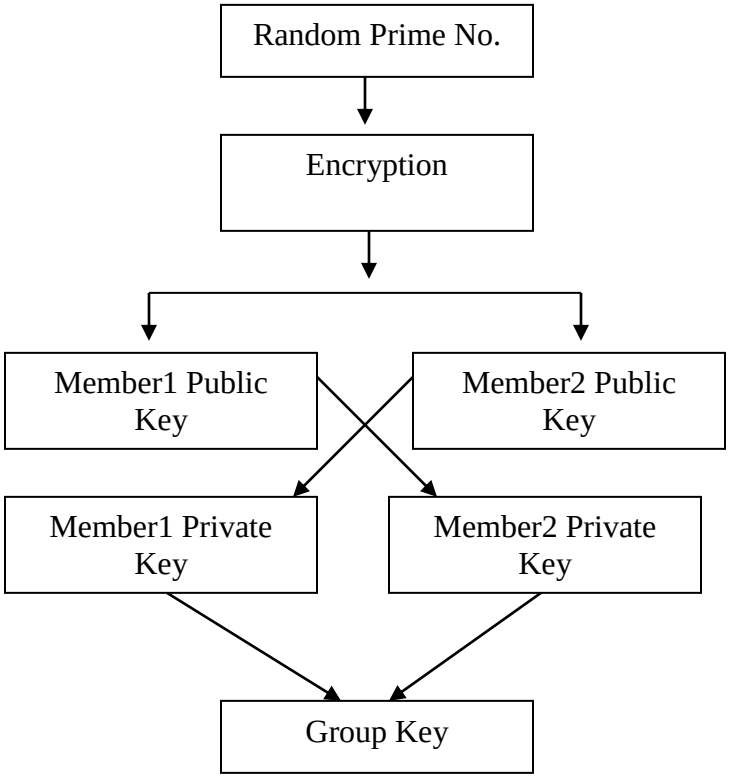

Server
Tree Structure
START
STOP
SETTINGS
Client List
Members
Public Key
BCT27
user1
2
Status

Data Flow Diagram

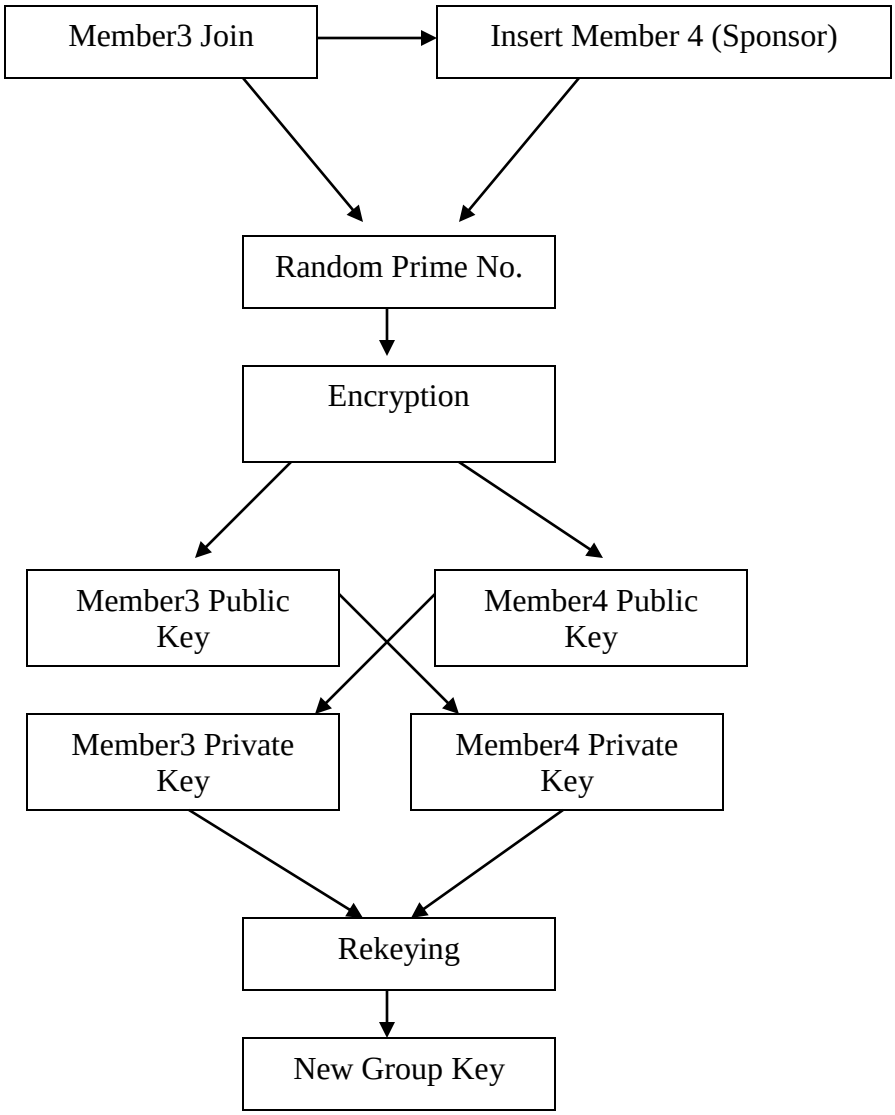


Architecture Design

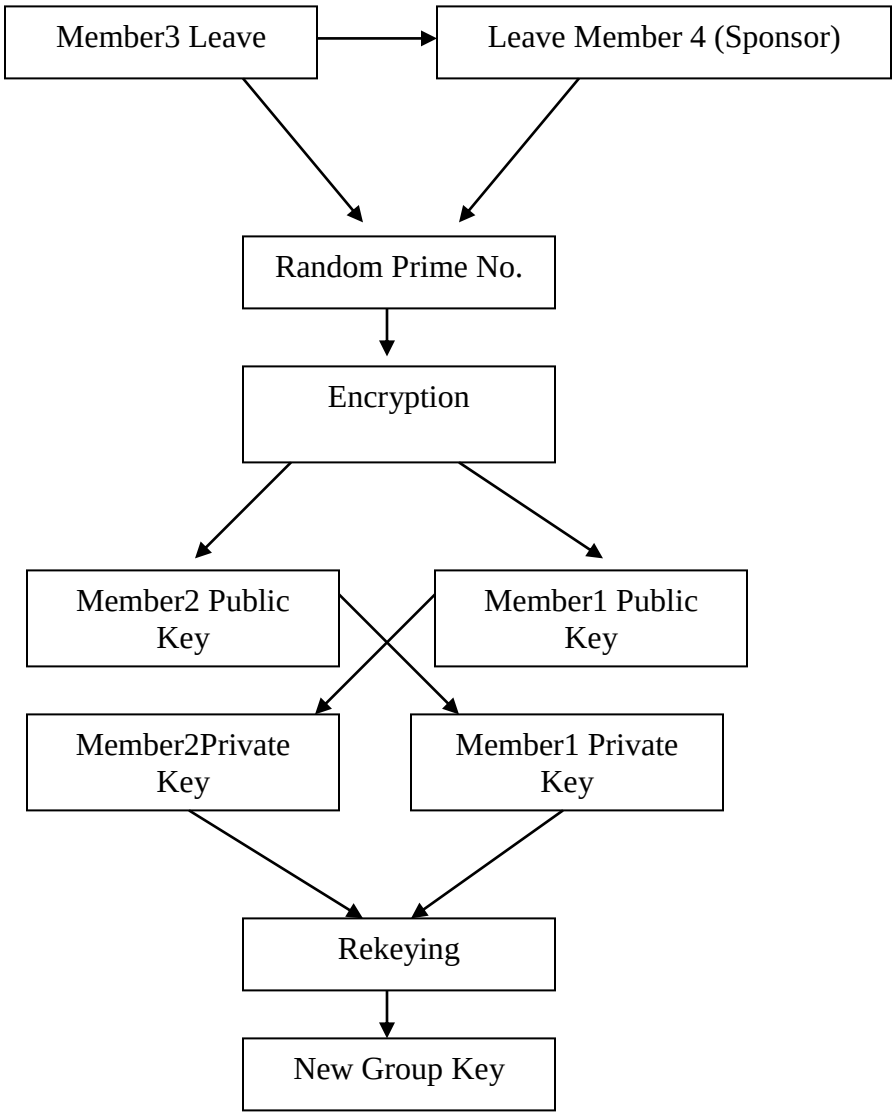
1. Group Key Generation



2. Rekeying the group key when a new member joins the workgroup



3. Rekeying the group key when a member leaves the workgroup



4. Sharing data within the workgroup