**Implementation**

**System components and relations**. The cloud computing storage model considered in this work consists of three main components :

(i) A data owner that can be an organization generating sensitive data to be stored in the cloud and made available for controlled external use and authorized users – a set of owner’s clients who have

the right to access the remote data.

(ii) A trusted third party (TTP), an entity who is trusted by all other system components, and has expertise and capabilities to detect and specify dishonest parties.

(iii) A CSP who manages cloud servers and provides paid storage space on its infrastructure to store the owner’s files and make them available for authorized users.

**Data Owner :**

The data owner organizes and generates sensitive data.

The data owner has to first login to his account and if he doesn’t have his account he has to register his account first into the Cloud Service Provider.

The C# code for registering the account is :

protected void User\_Register(object sender, EventArgs e)

{

DataTable dt = new DataTable();

dt = DataAccess.GetInstance().Cloud\_UserReg\_addCloudUserInfo(TxtName.Text.Trim(), TxtEmail.Text.Trim(), TxtPassword.Text.Trim(),TxtDOB.Text.Trim(), TxtPhone.Text.Trim(), TxtCity.Text.Trim(), TxtState.Text.Trim(), TxtCountry.Text.Trim());

if (dt.Rows.Count > 0)

{

if (dt.Rows[0]["ret"].Equals(1))

{

string User = TxtEmail.Text;

string[] words = User.Split('@');

string subPath = Server.MapPath("~/temp/" + words[0] + "/");

bool isExists = System.IO.Directory.Exists(subPath);

if (!isExists)

{

System.IO.Directory.CreateDirectory(subPath);

}

Utils.ShowAlertMessage("Registered Successfully");

Response.Redirect("~/CloudUserLogin.aspx");

}

else

{

Utils.ShowAlertMessage("Email have been registered already!!");

}

}

}

If the user is already registered he needs to login to his account.

C# code for login is:

protected void User\_Login(object sender, EventArgs e)

{

DataTable dt = new DataTable();

dt = DataAccess.GetInstance().CloudUserReg\_Authencate(TxtEmail.Text.Trim(), TxtPassword.Text.Trim());

if(dt.Rows.Count > 0)

{

Session["Email"] = TxtEmail.Text;

Response.Redirect("UploadFile.aspx");

}

else {

Utils.ShowAlertMessage("Incorrect Email and Password");

}

}

When the user login successfully then he can upload the file using the following implementation:

protected void BtnSubmit\_Click(object sender, EventArgs e)

{

string FileName = TxtFileName.Text.Trim();

uploadF up = new uploadF();

string UploadFile = up.savefile(FileUp);

int TPA = Convert.ToInt32(DropDownTPA.SelectedValue);

string Key = GetKeyGenerate(9);

string fileName = Path.GetFileNameWithoutExtension(UploadFile);

string fileExtension = Path.GetExtension(UploadFile);

string input = Server.MapPath("~/temp/") + fileName + fileExtension;

string output = Server.MapPath("~/temp/") + fileName + "\_enc" + fileExtension;

FileUp.SaveAs(input);

this.Encrypt(input, output, Key);

string uploadFilepath = fileName + "\_enc" + fileExtension;

string User = Session["Email"].ToString();

if (DataAccess.GetInstance().Files\_add\_File(User, TPA, FileName, uploadFilepath, Key))

{

Utils.ShowAlertMessage("File Uploaded Successfully");

TxtFileName.Text = "";

DropDownTPA.ClearSelection();

}

else

{

Utils.ShowAlertMessage("Server Down Try Later");

}

}

The user can also do the following work:

1. Can view all the approved files by the TTP as well as the pending files.
2. Can send request as well as accept the request for the respective file from other cloud users or TTP respectively.

protected void GridFileView\_RowCommand(object sender, GridViewCommandEventArgs e)

{

if (e.CommandName.Equals("Accept"))

{

if (DataAccess.GetInstance().TPARequest\_Update\_TPA\_Request(Convert.ToInt32(e.CommandArgument)))

{

MailSending mail = new MailSending();

GridViewRow row = (GridViewRow)(((ImageButton)e.CommandSource).NamingContainer);

Label LblUserName = (Label)row.Cells[0].FindControl("LblUserName");

HiddenField HiddenKey = (HiddenField)row.Cells[1].FindControl("HiddenKey");

Label LblFileName = (Label)row.Cells[2].FindControl("LblFileName");

Label LblEmail = (Label)row.Cells[3].FindControl("LblEmail");

HiddenField HiddenCloudUserName = (HiddenField)row.Cells[3].FindControl("HiddenCloudUserName");

string body = "Cloud User : " + HiddenCloudUserName.Value.ToString() + "\\n File Name : " + LblFileName.Text + "\\n secret Key : " + HiddenKey.Value.ToString();

mail.SendMail(LblEmail.Text, body, "Cloud User Accepted File Request");

LoadGrid();

Utils.ShowAlertMessage("Key has been sent to TPA");

}

}

}

1. Can update the pre uploaded file as well as update his own personal information.

protected void GridFileView\_RowCommand(object sender, GridViewCommandEventArgs e)

{

if (e.CommandName.Equals("Edit"))

{

GridViewRow row = (GridViewRow)(((ImageButton)e.CommandSource).NamingContainer);

HiddenField HiddenKey = (HiddenField)row.Cells[0].FindControl("HiddenKey");

Label lblTPANAme = (Label)row.Cells[0].FindControl("LblTPAName");

Label LblFileName = (Label)row.Cells[0].FindControl("LblFileName");

Label LblEmail = (Label)row.Cells[0].FindControl("LblEmail");

HiddenField HiddenCloudUserName = (HiddenField)row.Cells[0].FindControl("HiddenCloudUserName");

Response.Redirect("EditFileDetails.aspx?ID=" + e.CommandArgument + "&Key=" + HiddenKey.Value + "&TPAName=" + lblTPANAme.Text + "&FileName=" + LblFileName.Text + "&Email=" + LblEmail.Text + "&UserName=" + HiddenCloudUserName.Value);

}

}

**Trusted Third Party (TTP):**

It is an entity who is trusted by all other system components, and has expertise and capabilities to detect and specify dishonest parties.

TTP has to login in to his account and if his is new to network then he has to register his account and the CSP has to approve his registration.

TPA login code :

protected void TPA\_Login(object sender, EventArgs e)

{

DataTable dt = new DataTable();

dt = DataAccess.GetInstance().TPA\_Authenticate(TxtUserName.Text.Trim(), TxtPassword.Text.Trim());

if (dt.Rows.Count > 0)

{

Session["TPAEmail"] = TxtUserName.Text;

Response.Redirect("TPAGetFiles.aspx");

}

else

{

Utils.ShowAlertMessage("Invalid Username and Password");

}

}

The TTP has following work to do :

1. Has to approve the fresh uploaded files by the cloud user as well as approve the files updated by the cloud user and after verifying the respective file it again uploads the same file and which becomes trusted data as well as visible to other users too.

public void LoadGrid()

{

string Email = Session["TPAEmail"].ToString();

DataTable dt = new DataTable();

dt = DataAccess.GetInstance().Files\_Get\_TPA\_Files(Email);

if (dt.Rows.Count > 0)

{

GridFileView.DataSource = dt;

GridFileView.DataBind();

}

}

protected void GridFileView\_RowCommand(object sender, GridViewCommandEventArgs e)

{

if (e.CommandName.Equals("Send"))

{

if (DataAccess.GetInstance().TPARequest\_add\_TPA\_Request(Convert.ToInt32(e.CommandArgument)))

{

LoadGrid();

Utils.ShowAlertMessage("Request Sent");

}

}

}

1. TTP can send the request for the files which he need to verify and receive the decryption key of that file via email.

Files requested by TPA :

public void View\_request\_files()

{

string Email = Session["TPAEmail"].ToString();

DataTable dt = new DataTable();

dt = DataAccess.GetInstance().Files\_Get\_TPA\_Requsted\_Sent\_Files(Email);

if (dt.Rows.Count > 0)

{

GridFileView.DataSource = dt;

GridFileView.DataBind();

}

}

File request received by TPA:

public void Request\_received()

{

string Email = Session["TPAEmail"].ToString();

DataTable dt = new DataTable();

dt = DataAccess.GetInstance().Files\_Get\_TPA\_Requst\_Accepted\_Files(Email);

if (dt.Rows.Count > 0)

{

GridFileView.DataSource = dt;

GridFileView.DataBind();

}

}

protected void GridFileView\_RowCommand(object sender, GridViewCommandEventArgs e)

{

if (e.CommandName.Equals("Download"))

{

Response.Redirect("EnterKeyToDownload.aspx?File\_Id=" + Convert.ToInt32(e.CommandArgument));

}

}

**Cloud Service Provider (CSP):**

A CSP who manages cloud servers and provides paid storage space on its infrastructure to store the owner’s files and make them available for authorized users.

The CSP has following work to do :

1. CSP can check the user registration as well as can approve the user to avail the cloud facility or reject the user if he does not meet certain conditions and can also view all the registered users on the Cloud and can remove them if their term period is over.

Code for approving a new user:

protected void grvADview\_RowCommand(object sender, GridViewCommandEventArgs e)

{

if (e.CommandName.Equals("Approve"))

{

GridViewRow row = (GridViewRow)(((ImageButton)e.CommandSource).NamingContainer);

HiddenField Email = (HiddenField)row.Cells[0].FindControl("hndemail");

if (DataAccess.GetInstance().Cloud\_UserReg\_Approve\_Cloud\_User(Convert.ToInt32(e.CommandArgument)))

{

MailSending mail = new MailSending();

mail.SendMail(Email.Value,"You Have Sucessfully Registered to Cloud","Cloud Service Provider");

Utils.ShowAlertMessage("User Approved");

Response.Redirect("CheckUserReg.aspx");

}

}

1. Same as the procedure for the cloud users the CSP can do the same with the TTP registration approval as well as view all verified TTP.

Code for approving the TTP:

protected void Approve\_TTP(object sender, GridViewCommandEventArgs e)

{

if (e.CommandName.Equals("Approve"))

{

GridViewRow row = (GridViewRow)(((ImageButton)e.CommandSource).NamingContainer);

Label Email = (Label)row.Cells[0].FindControl("LblEmail");

if (DataAccess.GetInstance().Cloud\_TPA\_Approve(Convert.ToInt32(e.CommandArgument)))

{

MailSending mail = new MailSending();

mail.SendMail(Email.Text, "You Have Sucessfully Registered to Cloud", "Cloud Service Provider");

Utils.ShowAlertMessage("TPA Approved");

Response.Redirect("CheckTPA.aspx");

}

}

}

1. CSP can see list of all the files which are uploaded on his cloud server and by which user. He can also view the list of all the files which are pending by the TTP to be approved as well as see the list of files which are still pending to be approved by the TTP.

Code for the viewing files:

protected void GridFileView\_RowCommand(object sender, GridViewCommandEventArgs e)

{

if (e.CommandName.Equals("Download"))

{

GridViewRow row = (GridViewRow)(((ImageButton)e.CommandSource).NamingContainer);

HiddenField HiddenKey = (HiddenField)row.Cells[0].FindControl("HiddenKey");

string pname = e.CommandArgument.ToString();

string fileName = Path.GetFileNameWithoutExtension(pname);

string fileExtension = Path.GetExtension(pname);

string input = Server.MapPath("~/temp/") + fileName + fileExtension;

string output = Server.MapPath("~/temp/") + fileName + "\_dec" + fileExtension;

if ((System.IO.File.Exists(output)))

{

}

else

{

this.Decrypt(input, output, HiddenKey.Value);

}

Response.ContentType = "text/txt";

Response.AppendHeader("Content-Disposition", "attachment; filename=" + pname);

Response.TransmitFile(output);

Response.End();

if (output != null || output != string.Empty)

{

if ((System.IO.File.Exists(output)))

{

System.IO.File.Delete(output);

}

}

}

**Encryption and Decryption of the files:**

We are providing each file with a unique ID which includes the time (HH:MM:SS) n date (DD/MM/YYYY) at the moment when the file is uploaded which keep all the files unique from each other.

Encryption algorithm that we used is AES using a key length as 9 which is alpha-numeric, which is generated using a key generation algorithm.

**Encryption code:**

private void Encrypt(string inputFilePath, string outputfilePath, string EncryptionKey)

{

//string EncryptionKey = "MAKV2SPBNI99212";

using (Aes encryptor = Aes.Create())

{

Rfc2898DeriveBytes pdb = new Rfc2898DeriveBytes(EncryptionKey, new byte[] { 0x49, 0x76, 0x61, 0x6e, 0x20, 0x4d, 0x65, 0x64, 0x76, 0x65, 0x64, 0x65, 0x76 });

encryptor.Key = pdb.GetBytes(32);

encryptor.IV = pdb.GetBytes(16);

using (FileStream fsOutput = new FileStream(outputfilePath, FileMode.Create))

{

using (CryptoStream cs = new CryptoStream(fsOutput, encryptor.CreateEncryptor(), CryptoStreamMode.Write))

{

using (FileStream fsInput = new FileStream(inputFilePath, FileMode.Open))

{

int data;

while ((data = fsInput.ReadByte()) != -1)

cs.WriteByte((byte)data);

}

}

}

}

}

**Key Generation code:**

public string Key\_Generate(int length)

{

char[] chars = "1234567890ABCDEFGHIJKLMNOPQRSTUVWXYZ".ToCharArray();

string password = string.Empty;

Random random = new Random();

for (int i = 0; i < length; i++)

{

int x = random.Next(1, chars.Length);

if (!password.Contains(chars.GetValue(x).ToString()))

password += chars.GetValue(x);

else

i--;

}

return password;

}

**Decryption code:**

Decrypt method accepts the key from the User/TTP sent to him via email and uses the same key to decrypt the specific file.

Code to decrypt the file:

private void Decrypt(string inputFilePath, string outputfilePath, string EncryptionKey)

{

using (Aes encryptor = Aes.Create())

{

Rfc2898DeriveBytes pdb = new Rfc2898DeriveBytes(EncryptionKey, new byte[] { 0x49, 0x76, 0x61, 0x6e, 0x20, 0x4d, 0x65, 0x64, 0x76, 0x65, 0x64, 0x65, 0x76 });

encryptor.Key = pdb.GetBytes(32);

encryptor.IV = pdb.GetBytes(16);

using (FileStream fsInput = new FileStream(inputFilePath, FileMode.Open))

{

using (CryptoStream cs = new CryptoStream(fsInput, encryptor.CreateDecryptor(), CryptoStreamMode.Read))

{

using (FileStream fsOutput = new FileStream(outputfilePath, FileMode.Create))

{

int data;

while ((data = cs.ReadByte()) != -1)

{

fsOutput.WriteByte((byte)data);

}

}

}

}

}

}

**Method to trigger the Email (**using a fixed email id**):**

This method helps in generating and triggering the email when the file request is accepted but user to download the file or to verify the file, using the fixed email id i.e. “[bits.clouds@gmail.com](mailto:bits.clouds@gmail.com)” .

public void SendMail(string emailiduser, string Body, string subjecttext)

{

string emailid = emailiduser;//to user

string Subject = subjecttext;

string mailtext = Body;

System.Net.Mail.MailMessage msg = new System.Net.Mail.MailMessage();

msg.To.Add(emailid);

// msg.To.Add(emailidinfo);

msg.From = new MailAddress("bits.clouds@gmail.com", Subject, System.Text.Encoding.UTF8);

msg.Subject = Subject;

msg.SubjectEncoding = System.Text.Encoding.UTF8;

msg.Body = mailtext;

msg.BodyEncoding = System.Text.Encoding.UTF8;

msg.IsBodyHtml = true;

msg.Priority = MailPriority.High;

//Add the Creddentials

SmtpClient client = new SmtpClient();

client.Credentials = new System.Net.NetworkCredential("bits.clouds@gmail.com", "4ni10cs045");

client.Port = 25;//or use 587

client.Host = "smtp.gmail.com";

client.EnableSsl = true;

object userState = msg;

try

{

client.Send(msg);

}

catch (Exception ex)

{

}

}

**Method for File Upload and Save in the cloud server:**

This method helps in uploading different types of files and also approves which type of files can be uploaded into the cloud server.

public string savefile(FileUpload FileUpload)

{

string fname = "";

if (FileUpload.HasFile)

{

string filename = DateTime.Now.ToString("MMddyyyyHHmmss");

filename += Path.GetFileName(FileUpload.FileName);

string extension = System.IO.Path.GetExtension(filename);

try

{

if (extension == ".txt" || extension == ".doc" || extension == ".DOC" || extension == ".docx" || extension == ".DOCX" || extension == ".TXT")

{

FileUpload.SaveAs(System.Web.HttpContext.Current.Server.MapPath("~/temp/") + filename);

return filename;

}

}

catch (Exception ex)

{

Utils.ShowAlertMessage("Upload status: The file could not be uploaded.");

}

}

else

{

Utils.ShowAlertMessage("Select File to Upload");

}

return "file missing";

}