

**NET-BANKING SECURITY USING TWO-LAYER
CAPTCHA (Visual Cryptography)
TO AVOID ATTACKS**

Sravya Jarugu

ABSTRACT

The ability of hackers to infiltrate computer systems using computer attack programs and bots led to the development of captchas or completely Automated Public Turing Tests to set computers and humans apart. These captchas are usually encountered by the customers when they make any online transactions. The Text captcha is the most popular captcha scheme given its ease of construction and user friendliness. However, the next generation of hackers and programmers has decreased the expected security of these mechanisms, leaving websites open to attack. Text captchas are still widely used because it is believed that the attack speeds are slow, typically two to five seconds per image, and this is not seen as a critical threat. With deep learning techniques, there is a high success rate in breaking the roman character-based text captchas deployed by the top 50 most popular international websites and three chinese captchas that use a larger character set. These targeted schemes cover almost all existing resistance mechanisms. To overcome, this project proposes image-based captcha. A cryptographic technique based on visual secret sharing is used for image encryption. Using k out of n (k, n) visual secret sharing scheme a secret image is encrypted in shares which are meaningless images that can be transmitted or distributed over an untrusted communication channel. Also, a lot of customer sensitive data is exposed to the third party during online transactions. This project presents a new approach for providing only limited information that is necessary for online fund transfer during transfer thereby safeguarding customer data and increasing customer confidence by preventing identity theft. This project targets to tackle one of the most common and widely seen cybercrimes on web – Phishing.

Table of Contents

Introduction.....	4
Requirements	7
Proposed System.....	9
Look and Feel with Step-by-step procedure	12
Conclusion and Future Scope	16
References.....	17
Code	18

INTRODUCTION

Overview

In this paper, a new method is proposed, that uses text-based steganography and visual cryptography, which minimizes information sharing between consumer and online merchant but enable successful fund transfer from consumer's account to merchant's account thereby safeguarding consumer information and preventing misuse of information at merchant side. The method proposed is specifically for E-Commerce but can easily be extended for online as well as physical banking

Identity theft is the stealing of someone's identity in the form of personal information and misuse of that information. Identity theft and phishing are the common dangers of online transactions. Phishing is a criminal mechanism that employs both social engineering and technical subterfuge to steal consumers' personal identity data and financial account credentials. Financial and Retail Service are the most targeted industrial sectors of phishing attacks. Secure Socket Layer (SSL) encryption prevents the interception of consumer information in transit between the consumer and the online merchant. However, one must still trust merchant and its employees not to use consumer information for their own purchases and not to sell the information to others.

Covert communication is one of the most major concerns in today's world. With the growing needs of the secure means to transfer an information via Internet, the process of exchanging information secretly has become valuable due to the increase of data to be exchanged over Internet. Hence, the confidentiality and integrity of data requiring protection of unauthorized access and use of wanton, has led to tremendous growth in the field of data hiding. Systems for protection are divided to steganography which hides or Cryptography which encrypts the information or a mixture of both.

Hiding of such that the detection of messages gets impractical is the art of Steganography. Vast array of secret communication methods can be used to conceal the existence of a message.

Data can be hidden either using microdots, character arrangement, digital signatures, invisible inks, covert channels. Using Steganography, information can be hidden in various mediums known as carriers. The carriers can be images, audio files, video files and text files. On the other hand, Cryptography is derived from the Greek words: “kryptós” meaning "hidden" and “gráphein” meaning "to write" - or "hidden writing". It refers to the art of maintaining the secrecy of data by converting it to some other form.

Steganography by itself does not ensure secrecy, but neither does simple encryption. If these methods are combined, however, stronger encryption methods result. Security is achieved when cryptography and steganography are solely used, but in a wide area network such as Internet, it does not withstand security attacks. Hence, Steganography and Cryptography can be combined [4] to hide a secret message. (i.e. If an encrypted message is intercepted, the interceptor will know that the text is an encrypted message. But the interceptor may not know that a hidden message even exists with steganography). This provides two levels of security and with the usage of encryption and combining hashes, integrity and authentication can be achieved.

The main objective of this project is to safeguard customer data and prevent phishing attack during online shopping by using visual cryptography and steganography [1]. Proposes a combined image-based steganography [1] and visual cryptography authentication system for customer authentication in net banking. The use of images is explored to preserve the privacy of image captcha by decomposing the original image captcha into two shares that are stored in separate database servers such that the original image captcha can be revealed only when both are combined. The individual sheet images do not reveal the identity of the original image captcha. Once the original image captcha is revealed to the user it can be used as a password.

Text Steganography

In this method in the nth letter of every word of a text message the secret message is hidden. Due to the boom in the internet and different types of digital files its importance has decreased. Text files have a very small amount of data hence text steganography using digital files is not used very often.

Audio Steganography

In this method in a medium such as an audio file, the secret message is hidden in an imperceptible manner. The characteristics of stego message that is derived after steganography will be same as the and the host message that is present before steganography. There are a variety of techniques for embedding messages into digital, but it is a very difficult process and is very expensive.

Image Steganography

In this type of steganography, the secret message is hidden inside a digital image using embedding algorithms and a secret key. Stego image which is the resultant is sent to the receiver. It is then processed by the extraction algorithm using the same key. This stego-image when sent from a sender to the receiver on the other end, an unauthorized person can only notice the transmission of an image but the existence of the hidden message inside the image cannot be guessed. Video Steganography is a technique in which the secret message is hidden inside a video file.

REQUIREMENTS

Hardware Requirements

Processor: Pentium IV 2.4 GHz

Hard Disk: 40 GB

RAM: 256 MB

Software Requirements

Operating System: Windows 7 or above

Front End: C#, ASP.Net

Back End: SQL Server 2005

IDE: Microsoft Visual Studio

Software Description

Microsoft Visual Studio

Microsoft Visual Studio is an integrated development environment from Microsoft. It is used to develop computer programs for Microsoft Windows, as well as web sites, web applications and web services. Visual Studio uses such as Windows API, Windows Forms, Windows Presentation Foundation.

Visual Studio includes a code editor supporting IntelliSense (the code completion component). The integrated debugger works both as a source-level debugger and a machine-level debugger. Other built-in tools include a forms designer for building GUI applications, web designer, class designer, and database schema designer. It accepts plug-ins that enhances the functionality at almost every level which includes adding support for source-control systems.

C# .NET

C# is one of many .NET programming languages. It is object-oriented and allows you to build reusable components. The Microsoft .NET framework is a software framework developed by Microsoft that runs primarily on Microsoft Windows.

Features of .NET

Microsoft .NET is a set of Microsoft software technologies for rapidly building and integrating XML Web services, Microsoft Windows-based applications, and Web solutions. The .NET Framework is a language-neutral platform for writing programs that can easily and securely interoperate. There's no language barrier with .NET: there are numerous languages available to the developer including Managed C++, C#, Visual Basic and Java Script. The .NET framework provides the foundation for components to interact seamlessly, whether locally or remotely on different platforms.

The .NET Framework has two main parts:

1. The Common Language Runtime (CLR).
2. A hierarchical set of class libraries.

The CLR is described as the “execution engine” of .NET. It provides the environment within which programs run. The most important features are Conversion from a low-level assembler-style language, called Intermediate Language (IL), into code native to the platform being executed on. The .NET Framework is a new computing platform that simplifies application development in the highly distributed environment of the Internet. It aims

1. To provide a consistent object-oriented programming environment whether object codes is stored and executed locally on Internet.
2. To provide a code-execution environment to minimizes software deployment and guarantees safe execution of code and eliminate performance problems.

SQL Server

To create a database determines the name of the database, its owner (the user who creates the database), its size and the file groups used to store it. Microsoft SQL Server is a relational database management system developed by Microsoft. As a database, it is a software product whose primary function is to store and retrieve data as requested by other software applications, be it those on the same computer or those running on another computer across a network (including the Internet). There are at least a dozen different editions of Microsoft SQL Server aimed at different audiences and for workloads ranging from small single-machine applications to large Internet-facing applications with many concurrent users.

PROPOSED SYSTEM

The proposed system makes use of images to provide security. A cryptographic technique which is based on visual secret sharing is used for image encryption. Using k out of n (k, n) visual secret sharing scheme, a secret image is encrypted in shares which are meaningless images that can be transmitted or distributed over an untrusted communication channel. In this proposal the image is divided into two shares only. Combining these k shares or more will give the original secret image. The use of images is explored to preserve the privacy of image captcha by decomposing the original image captcha into TWO shares that are stored in separate database servers such that the original image captcha can be revealed only when both are combined. Individual sheet images do not reveal the identity of the original image captcha. Once the original image captcha is revealed to the user it can be used as the password.

Image Generation

When the customer logs-in for the first time, a text captcha is generated which is embedded into the image and this image is then divided into two shares.

Algorithm [2] for embedding the text in the image:

1. Convert the text captcha into binary format by substituting the binary equivalent for every letter.
2. Generate the Fibonacci number for the binary converted captcha.
3. If the number $|2| = 0$ then XOR message 1st bit to White Color and 2nd bit to Black Color of Image pixel.
If number $|2| = 1$ then XOR message 1st bit to Black color and 2nd bit to White color of Image pixel.
4. Go to next pixel of the image and go to next bit of the message.
5. Repeat steps 3 and 4 until all bits of the message are embedded into Image.

The secret image is divided into two shares so that reconstruction of an image from a share is impossible. Each share is printed in transparency. A pixel in the image is divided into two sub pixels depending on whether the pixel is black or white. If the original image pixel was black, the

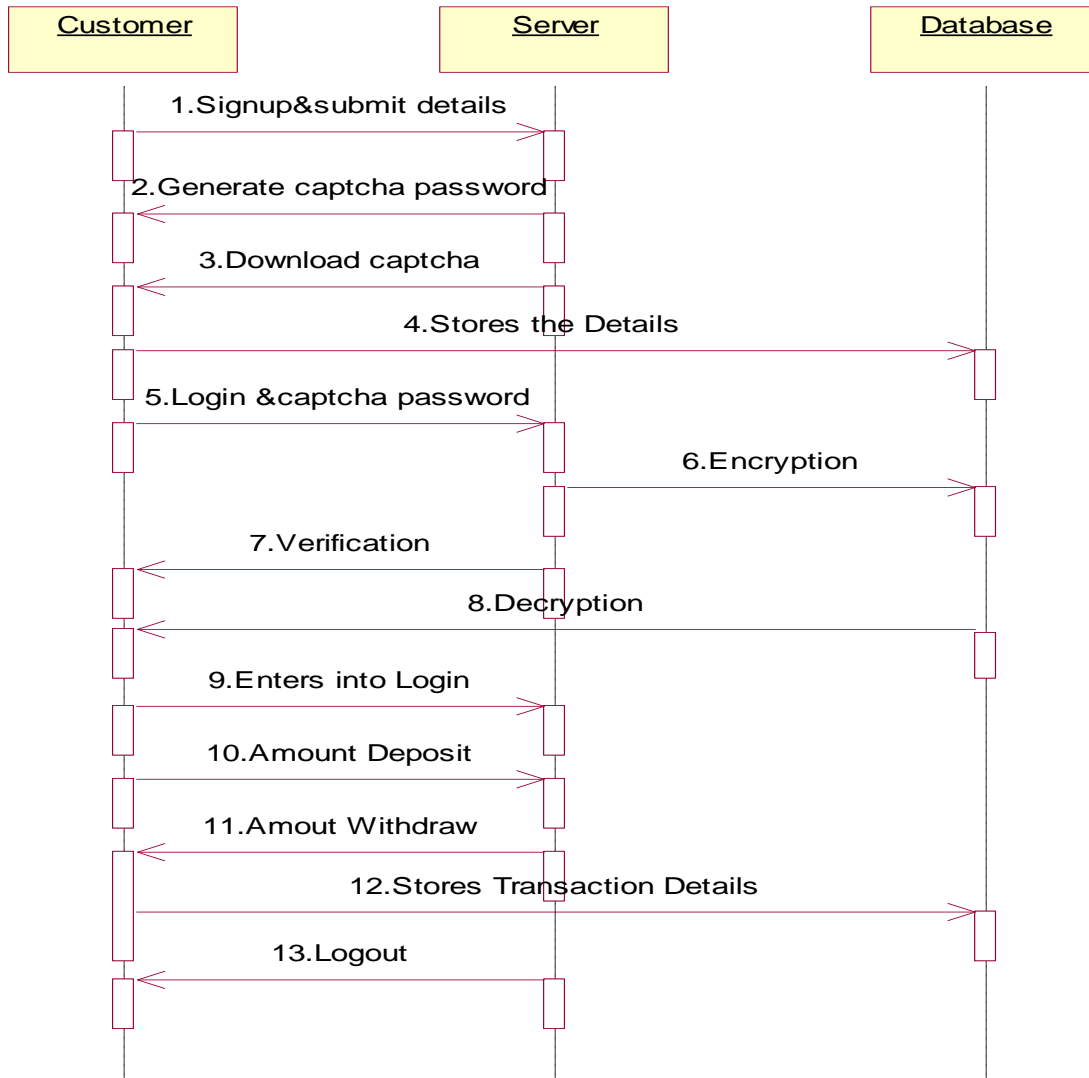
pixel pairs in the component images must be complementary; randomly shade one ■□, and the other □■ [3]. When these complementary pairs are overlapped, they will appear dark gray. On the other hand, if the original image pixel was white, the pixel pairs in the component images must match: both ■□ or both □■ [3]. When these matching pairs are overlapped, they will appear light gray. So, when the two component images are superimposed, the original image appears.

Pixel	White □	Black ■
Prob.	50% 50%	50% 50%
Share 1	■□ □■	■□ □■
Share 2	■□ □■	□■ □■
Stack share 1 & 2	■□ □■	■□ ■□

Steps to log-in to the Net-Banking

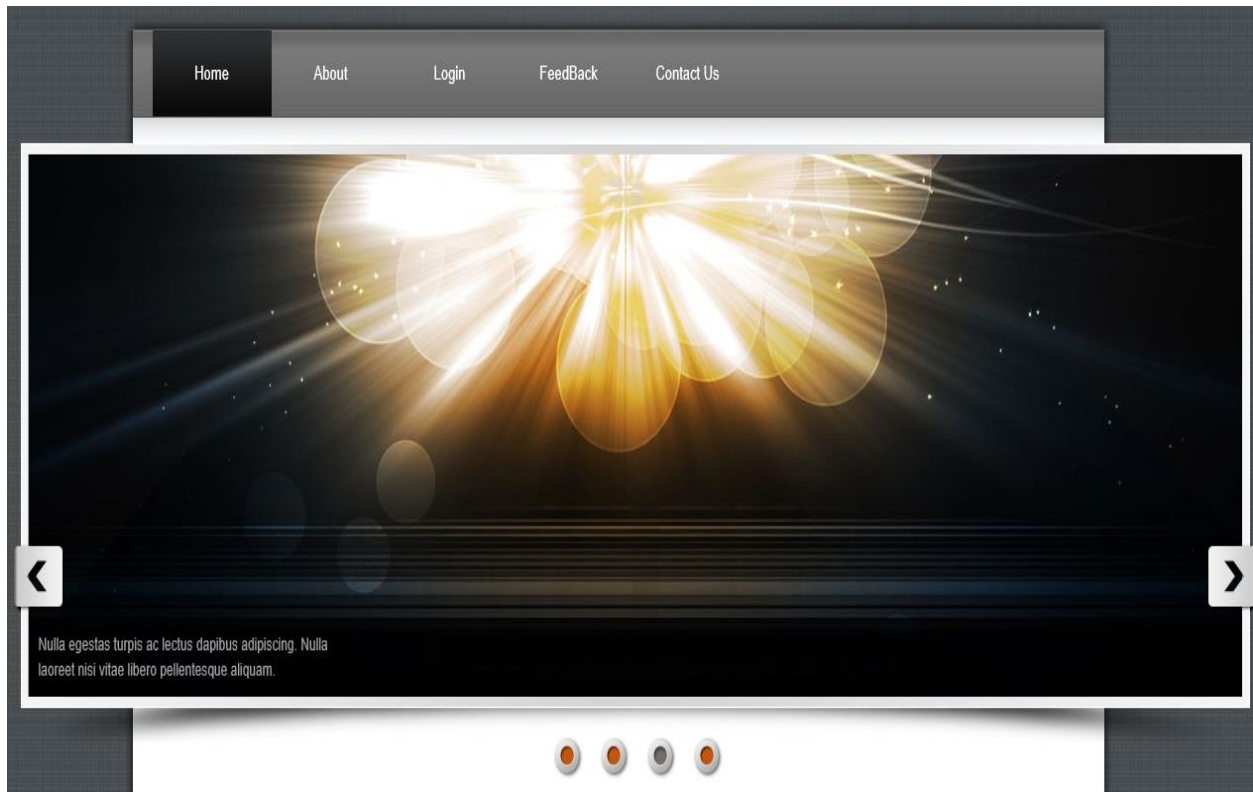
1. Sign-up to the Net Banking account using the username and password given by the bank.
For the first time when the customer logs in to the account an image is generated and divided into two parts.
2. A link to download the image (one-half of the image) will be displayed. This image is to be download the image and is to be kept safe. The other half of the image will be stored in the database of the bank.
3. The customer when tries to log-in to the net banking account with the password created, he is supposed to upload the downloaded image. After uploading, the other half of the image is pulled from the database and combined with the image uploaded by the customer and if this complete image is same as the original image generated when the customer has first logged-in then the transaction will be successful, otherwise the transaction will be unsuccessful.
4. This image when opened will have no content in it. This way security is ensured, provided, customer does not share the image with anybody.

Sequence Diagram



Sequence diagram models the flow of logic within the system in a visual manner, enabling you to document and validate the logic. It is commonly used for both analysis and design purpose. Sequence diagram are the most popular UML artifact for dynamic modeling, which focuses on identifying the behavior within your system.

LOOK AND FEEL WITH Step-by-step PROCEDURE




This web application can be logged-in by an admin and a user (customer). Admin is responsible for generating a username and the password to the customer when they open an account in the bank. This password will be shared with the customer using which he can log-in for the first time.

A screenshot of the "Admin Login" form. At the top of the form, there is a 3D illustration of a white character sitting on a red chair, using a laptop. To the right of the illustration, the text "Admin Login" is displayed in a stylized, orange font. Below the illustration and text, there are two input fields. The first field is labeled "User Id:" and the second field is labeled "Password:". Below the "Password:" field, there is a button labeled "Login".

User Login Page

USER LOGIN HERE	
	
<input type="text"/>	
Label	
Username:	<input type="text"/>
Password:	<input type="password"/>
BrowsercaptchaImage:	<input type="text"/> <input type="button" value="Browse..."/> <input type="button" value="Show Images"/>
Servercaptcha:	
Captcha:	

User Sign-up page, Sign-up using the password generated by admin

CREATE YOUR LOGIN	
All Fields are required	
Username:	<input type="text" value="sravya"/>
Password:	<input type="password" value="....."/>
ReEnter Password:	<input type="password" value="....."/>
E-Mail Address:	<input type="text" value="shgd@gmail.com"/>
Enter The Captcha Text:	<input type="text" value="b2K65"/> <input type="button" value="X"/>
	
<input type="button" value="Submit"/>	<input type="button" value="Login"/>

Goto Login

USER DETAILS

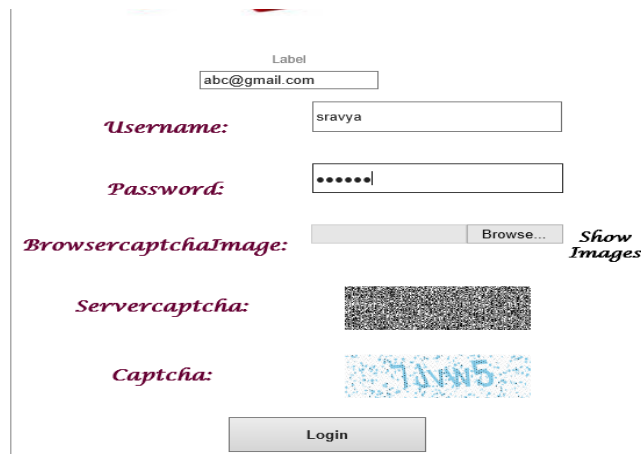
Download

* You have to upload this file where you Login

Download your Captcha

CAPTCHA COMPARISON

Once the user has signed in, he will be re-directed to page with a download link for the image. This image is to be downloaded and kept safe.

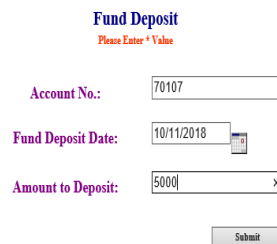


The login form is enclosed in a light gray border. It contains the following elements: a 'Label' input field with 'abc@gmail.com'; a 'Username:' label followed by an input field with 'sravya'; a 'Password:' label followed by a masked input field with six dots; a 'BrowsercaptchaImage:' label followed by a 'Browse...' button; a 'Servercaptcha:' label followed by a noisy grayscale image; a 'Captcha:' label followed by a blue-tinted image with the text '7JW5'; and a 'Login' button at the bottom center. A 'Show Images' link is positioned to the right of the 'BrowsercaptchaImage' section.

The user should upload (one-half) of the image, enter the username, password and click on “Show Images” to login. When clicked on “Show Images”, the other half of the image is pulled from the database, combined and validated. User will be logged-in only if the combined image matches with the original image.



On a successful log-in, the user can now transfer through fund transfer



The 'Fund Deposit' form has a blue title and a red instruction 'Please Enter * Value'. It includes three input fields: 'Account No.' with '70107', 'Fund Deposit Date' with '10/11/2018' and a calendar icon, and 'Amount to Deposit' with '5000' and a clear 'x' button. A 'Submit' button is located at the bottom right.

Enter the account number and the amount to be deposited

Master Manage Cheque Book Fund Deposit Fund Withdrawal Customer Account Sign Out

Fund Deposit

Select Date: 10/11/2018 Show

..... ↑

When the user has requested for a fund deposit, Admin will be notified regarding the fund transfer request on the day the transaction is made. Only the Admin has the right to approve. Once approved, the amount will be deposited to the respective account.

Master Manage Cheque Book Fund Deposit Fund Withdrawal Customer Account Sign Out

Fund Deposit

Select Date: 10/11/2018 Show

Account No.	Amount	
70107	5000.0000	Select

..... ↑

Admin can see the Fund transfer requests made by the customers on a day by selecting the date.

Master Manage Cheque Book Fund Deposit Fund Withdrawal Customer Account Sign Out

Fund Deposit To Account

Please Enter * Value

Fund Transfer Date: 10/11/2018

To Account No: 70107

Amount To Deposit: 5000.0000

Deposit Amount

Fund Transferred Successfully To: 70107

..... ↑

Admin will approve the request and deposit the amount into the respective account.

CONCLUSION AND FUTURE SCOPE

Two Layer security is ensured by embedding the text captcha into an image and diving it into two shares where each share alone doesn't reveal any information of another. Visual cryptography combined with steganography helped to achieve this. Despite its complexity, it has some drawbacks which leave scope for future development.

1. There is a problem of pixel expansion where a Pixel in the image when divided into 2 sub pixels increase the width of the entire image and thus there will be increase in bandwidth required and so increase in the power consumption. Development can be made to avoid this.
2. The same can be implemented for 3D images to achieve more security.
3. The complexity can be increased with the use of Digital Signatures instead of text captcha.
4. The one share of the image with the customer is to be kept safe and unaltered.

References

- [1] K. Bennet, “Linguistic Steganography: Surevey, Analysis, and Robustness Concerns for Hiding information in Text,” Purdue University, Cerias Tech Report 2004 - 2013.
- [2] <https://pdfs.semanticscholar.org/1ff3/e8fc93e4c11b7e0e8cb4d38c893bc71c710b.pdf> - Rohit Kumar Jain , Dr. Rahul Malhotra and Kulbushan Rassewatt, M. Tech. Student, Department of Electronics and Communication Engineering, Guru Teg Bahadur Khalsa Institute of Engineering & Technology, Chhapiawali, Malout, INDIA.
- [3] https://en.wikipedia.org/wiki/Visual_cryptography
- [4] <http://www.wseas.us/e-library/conferences/2014/Malaysia/ACACOS/ACACOS-17.pdf> - HAYFAA ABDULZAHRA, ROBIAH AHMAD1, NORLIZA MOHD NOOR Department of Engineering, UTM Razak School of Engineering and Advanced Technology, UTM Kuala Lumpur, 54100 Jalan Semarak, Kuala Lumpur Malaysia.
- [5] Cryptography and Network Security – Seventh Edition, By William Stallings.
- [6] <https://www.w3schools.com/html/> - for HTML
- [7] <https://www.geeksforgeeks.org/introduction-to-c-sharp/> - for C#

CODE

```
using System;
using System.Data;
using System.Configuration;
using System.Web;
using System.Web.Security;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
using System.Web.UI.HtmlControls;
using System.Data.SqlClient;
using ElectronicCash.DAL;
public class CustomerLoginBL : Connection
{
    public static DataSet ds;
    public CustomerLoginBL()
    {
    }
    private string uname, password;
    public string Uname
    {
        get { return uname; }
        set { uname = value; }
    }
    public string Password
    {
        get { return password; }
        set { password = value; }
    }
    public void InsertUserLoginInfo()
    {
        SqlParameter[] p = new SqlParameter[2];
```

```

        p[0] = new SqlParameter("@uname", this.uname);
        p[0].DbType = DbType.String;
        p[1] = new SqlParameter("@password", this.password);
        p[1].DbType = DbType.String;
        SqlHelper.ExecuteNonQuery(con, CommandType.StoredProcedure,
"SpAddUserLoginInfo", p);
    }
    public bool GetAccountHolder()
    {
        int count;
        SqlParameter[] p = new SqlParameter[2];
        p[0] = new SqlParameter("@uname", this.uname);
        p[0].DbType = DbType.String;
        p[1] = new SqlParameter("@password", this.password);
        p[1].DbType = DbType.String;
        count = int.Parse(SqlHelper.ExecuteScalar(con, CommandType.StoredProcedure,
"spCheckUser", p).ToString());
        if (count > 0)
            return true;
        else
            return false;
    }
    public bool GetAdmin()
    {
        int count;
        SqlParameter[] p = new SqlParameter[2];
        p[0] = new SqlParameter("@uname", this.uname);
        p[0].DbType = DbType.String;
        p[1] = new SqlParameter("@password", this.password);
        p[1].DbType = DbType.String; count = int.Parse(SqlHelper.ExecuteScalar(con,
CommandType.StoredProcedure, "spCheckAdmin", p).ToString());

```

```

        if (count > 0)
            return true;
        else
            return false;
    }
    public void ChangePassword()
    {
        SqlParameter[] p = new SqlParameter[2];
        p[0] = new SqlParameter("@uname", this.uname);
        p[0].DbType = DbType.String;
        p[1] = new SqlParameter("@password", this.password);
        p[1].DbType = DbType.String;

        SqlHelper.ExecuteNonQuery(con, CommandType.StoredProcedure, "SpChangePassword",
p);

    }
    public bool GetUserName(string uname)
    {
        int count = 0;
        string str = "select count(user_name) from login_master where user_name='" + uname + "'";
        count = int.Parse(SqlHelper.ExecuteScalar(con, CommandType.Text, str).ToString());
        if (count > 0)
        {
            return true;
        }
        else
        {
            return false;
        }
    }
}

```

```

using System;
using System.Collections;
using System.Configuration;
using System.Data;
using System.Linq;
using System.Web;
using System.Web.Security;
using System.Web.UI;
using System.Web.UI.HtmlControls;
using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
using System.Xml.Linq;
using System.IO;
using System.Data.SqlClient;
using System.Drawing.Imaging;
using System.Collections.Generic;
using System.ComponentModel;
using System.Drawing;
using System.Text;
public partial class new_user : System.Web.UI.Page
{
    SqlConnection con = new SqlConnection("Data Source=SHAHUL;Initial
Catalog=VisualCryptography;User Id=sa;Password=sa");
    SqlCommand cmd=new SqlCommand();
    string fn;
    int fs;
    static int uid;
    protected void Page_Load(object sender, EventArgs e)
    {
        Label12.Text = "";
    }
}

```

```

protected void Button1_Click(object sender, EventArgs e)
{
    if (TextBox5.Text == this.Session["CaptchaImageText"].ToString())
    {
        fn = Session["FName"].ToString();
        fs = Convert.ToInt32(Session["fs"]);
        string name, p, rp, eid;
        name = un.Text;
        p = passwd.Text;
        rp = reepasswd.Text;
        eid = email.Text;
        string doname;
        string ext = ".jpg";
        con.Open();
        cmd = new SqlCommand("select max(userid) from newuser", con);
        object result = cmd.ExecuteScalar();
        if (result.GetType() != typeof(DBNull))
        {
            uid = Convert.ToInt32(result);
        }
        else
        {
            uid = 0;
        }
        con.Close();
        uid = uid + 1;
        string fileName = Path.GetFileName(fn);
        doname = Path.GetFileName(fn);
        string fileExtension = Path.GetExtension(fn);
        string documentType = String.Empty;
        switch (fileExtension

```

```

{
    case ".gif":
        documentType = "image/gif";
        break;
    case ".png":
        documentType = "image/png";
        break;
    case ".jpg":
        documentType = "image/jpg";
        break;
}
int fileSize = Convert.ToInt32(fs);
Byte[] documentBinary = new Byte[fileSize];
con.Open();
cmd = new SqlCommand("insert into newuser values('" + name + "','" + p + "','" + rp + "','" +
eid + "','" + doname + "','" + ext + "','" + fileSize + "',@DocData)", con);
cmd.Parameters.AddWithValue("@DocData", documentBinary);
int res = cmd.ExecuteNonQuery();
if (res > 0)
{
    Label12.Text = "File Saved to Database";
    Response.Write("<script language='javascript'>alert('\nFile Saved to
Database');</script>");
}
con.Close();
Session["FName"] = doname.ToString();
Session["fs"] = fs;
Response.Redirect("~/downloadcaptcha.aspx");
}
else
{

```

```

        ClientScript.RegisterStartupScript(Page.GetType(), "validation", "<script
language='javascript'>alert('Captcha Image and Text are not same. Please Try
Again!!!')</script>");
    }
}
protected void Button2_Click(object sender, EventArgs e)
{
}
}
using System;
using System.Collections;
using System.Configuration;
using System.Data;
using System.Linq;
using System.Web;
using System.Web.Security;
using System.Web.UI;
using System.Web.UI.HtmlControls;
using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
using System.Xml.Linq;
using System.IO;
using System.Data.SqlClient;
public partial class downloadcaptcha : System.Web.UI.Page
{
    SqlConnection con = new SqlConnection("Data Source=SHAHUL;Initial
Catalog=VisualCryptography;User Id=sa;Password=sa");
    string fn;
    string fileName;
    int fs;
protected void Page_Load(object sender, EventArgs e)

```



```

fn = Session["FName"].ToString();
    fs = Convert.ToInt32(Session["fs"]);
}
protected void LinkButton1_Click(object sender, EventArgs e)
{
    SqlCommand cmd = new SqlCommand("select * from newuser where fname = '" + fn + "'
and sice = '" + fs + "'", con);
    DataTable dt = GetData(cmd);
    if (dt != null)
    {
        download(dt);
    }
    string fileLocCopy = @"C:\\jp" + fn;
    string fileLoc = Server.MapPath(@"~\Docs\" + fn);
    if (File.Exists(fileLoc))
    {
        if (File.Exists(fileLocCopy))
            File.Delete(fileLocCopy);
        File.Copy(fileLoc, fileLocCopy);
    }
}
private DataTable GetData(SqlCommand cmd)
{
    DataTable dt = new DataTable();
    SqlDataAdapter sda = new SqlDataAdapter();
    cmd.CommandType = CommandType.Text;
    cmd.Connection = con;
    try
    {
        con.Open();
        sda.SelectCommand = cmd;

```

```

        sda.Fill(dt);
        return dt;
    }
    catch
    {
        return null;
    }
    finally
    {
        con.Close();
        sda.Dispose();
        con.Dispose();
    }
}

private void download(DataTable dt)
{
    string ftype;
    Byte[] bytes = (Byte[])dt.Rows[0]["data"];
    Response.Buffer = true;
    Response.Charset = "";
    Response.Cache.SetCacheability(HttpCacheability.NoCache);
    ftype = dt.Rows[0]["type"].ToString();
    Response.ContentType = "application/" + ftype + " ";
    Response.AddHeader("content-disposition", "attachment;filename=" +
dt.Rows[0]["fname"].ToString());
    Response.BinaryWrite(bytes);
    Response.Flush();
    Response.End();
}
}

```