using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Windows.Forms;

using System.IO;

using System.Text;

using System.Security.Cryptography;

namespace MIMO

{

public partial class FileUplaod : Form

{

string filename;

string k;

public FileStream fs;

string mergeFolder;

List<string> Packets = new List<string>();

public static byte[] alicePublicKey;

public FileUplaod()

{

InitializeComponent();

}

private static void Send(byte[] key, string secretMessage, out byte[] encryptedMessage, out byte[] iv)

{

using (Aes aes = new AesCryptoServiceProvider())

{

aes.Key = key;

iv = aes.IV;

// Encrypt the message

using (MemoryStream ciphertext = new MemoryStream())

using (CryptoStream cs = new CryptoStream(ciphertext, aes.CreateEncryptor(), CryptoStreamMode.Write))

{

byte[] plaintextMessage = Encoding.UTF8.GetBytes(secretMessage);

cs.Write(plaintextMessage, 0, plaintextMessage.Length);

cs.Close();

encryptedMessage = ciphertext.ToArray();

}

}

}

private void button2\_Click(object sender, EventArgs e)

{

ECDiffieHellmanCng alice = new ECDiffieHellmanCng();

alice.KeyDerivationFunction = ECDiffieHellmanKeyDerivationFunction.Hash;

alice.HashAlgorithm = CngAlgorithm.Sha256;

alicePublicKey = alice.PublicKey.ToByteArray();

Bob bob = new Bob();

CngKey k = CngKey.Import(bob.bobPublicKey, CngKeyBlobFormat.EccPublicBlob);

byte[] aliceKey = alice.DeriveKeyMaterial(CngKey.Import(bob.bobPublicKey, CngKeyBlobFormat.EccPublicBlob));

byte[] encryptedMessage = null;

byte[] iv = null;

Send(aliceKey, "Secret message", out encryptedMessage, out iv);

bob.Receive(encryptedMessage, iv);

string path1 = Path.GetDirectoryName(Application.ExecutablePath).ToString();

string path2 = Path.GetDirectoryName(Application.ExecutablePath).ToString();

EncryptFile(path1 + @"\File\" + filename, path2 + @"\Encrypt\" + filename);

MessageBox.Show("Successfully");

}

private void button1\_Click(object sender, EventArgs e)

{

string path = Path.GetDirectoryName(Application.ExecutablePath).ToString();

//OpenFileDialog op = new OpenFileDialog();

//op.ShowDialog();

//textBox1.Text = op.FileName;

//filename = System.IO.Path.GetFileName(op.FileName);

//MessageBox.Show(filename);

string destination = path + @"\File\";

OpenFileDialog ofd = new OpenFileDialog();

ofd.Multiselect = true;

if (DialogResult.OK == ofd.ShowDialog())

{

textBox1.Text = ofd.FileName;

filename = System.IO.Path.GetFileName(ofd.FileName);

foreach (string file in ofd.FileNames)

{

File.Copy(file, Path.Combine(destination, Path.GetFileName(file)));

}

}

}

public bool SplitFile(string SourceFile, int nNoofFiles)

{

bool Split = false;

try

{

FileStream fs = new FileStream(SourceFile, FileMode.Open, FileAccess.Read);

int SizeofEachFile = (int)Math.Ceiling((double)fs.Length / nNoofFiles);

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

{

string baseFileName = Path.GetFileNameWithoutExtension(SourceFile);

string Extension = Path.GetExtension(SourceFile);

FileStream outputFile = new FileStream(Path.GetDirectoryName(SourceFile) + "\\" + baseFileName + "." +

i.ToString().PadLeft(3, Convert.ToChar("0")) + Extension + ".txt", FileMode.Create, FileAccess.Write);

//FileStream outputFile = new FileStream(Path.GetDirectoryName(SourceFile) + "\\" + baseFileName + "." +

// i.ToString().PadLeft(5, Convert.ToChar("0")) + Extension + ".txt", FileMode.Create, FileAccess.Write);

mergeFolder = Path.GetDirectoryName(SourceFile);

int bytesRead = 0;

byte[] buffer = new byte[SizeofEachFile];

if ((bytesRead = fs.Read(buffer, 0, SizeofEachFile)) > 0)

{

outputFile.Write(buffer, 0, bytesRead);

//outp.Write(buffer, 0, BytesRead);

string packet = baseFileName + "." + i.ToString().PadLeft(3, Convert.ToChar("0")) + Extension.ToString();

Packets.Add(packet);

}

outputFile.Close();

}

fs.Close();

}

catch (Exception Ex)

{

throw new ArgumentException(Ex.Message);

}

return Split;

}

private void EncryptFile(string inputFile, string outputFile)

{

string password = @"myKey123";

UnicodeEncoding UE = new UnicodeEncoding();

byte[] key = UE.GetBytes(password);

string cryptFile = outputFile;

FileStream fsCrypt = new FileStream(cryptFile, FileMode.Create);

RijndaelManaged RMCrypto = new RijndaelManaged();

CryptoStream cs = new CryptoStream(fsCrypt,

RMCrypto.CreateEncryptor(key, key),

CryptoStreamMode.Write);

FileStream fsIn = new FileStream(inputFile, FileMode.Open);

int data;

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

cs.WriteByte((byte)data);

fsIn.Close();

cs.Close();

fsCrypt.Close();

}

private void button3\_Click(object sender, EventArgs e)

{

string path2 = Path.GetDirectoryName(Application.ExecutablePath).ToString();

//EncryptFile(path1 + @"\File\" + filename, path2 + @"\Encrypt\" + filename);

SplitFile(path2 + @"\Encrypt\" + filename, Convert.ToInt32(4));

textBox2.Text = (Packets[0].ToString());

textBox3.Text = (Packets[1].ToString());

textBox4.Text = (Packets[2].ToString());

textBox5.Text = (Packets[3].ToString());

}

}

public class Bob

{

public byte[] bobPublicKey;

private byte[] bobKey;

public Bob()

{

using (ECDiffieHellmanCng bob = new ECDiffieHellmanCng())

{

bob.KeyDerivationFunction = ECDiffieHellmanKeyDerivationFunction.Hash;

bob.HashAlgorithm = CngAlgorithm.Sha256;

bobPublicKey = bob.PublicKey.ToByteArray();

bobKey = bob.DeriveKeyMaterial(CngKey.Import(FileUplaod.alicePublicKey, CngKeyBlobFormat.EccPublicBlob));

}

}

public void Receive(byte[] encryptedMessage, byte[] iv)

{

using (Aes aes = new AesCryptoServiceProvider())

{

aes.Key = bobKey;

aes.IV = iv;

// Decrypt the message

using (MemoryStream plaintext = new MemoryStream())

{

using (CryptoStream cs = new CryptoStream(plaintext, aes.CreateDecryptor(), CryptoStreamMode.Write))

{

cs.Write(encryptedMessage, 0, encryptedMessage.Length);

cs.Close();

string message = Encoding.UTF8.GetString(plaintext.ToArray());

Console.WriteLine(message);

}

}

}

}

private void button2\_Click(object sender, EventArgs e)

{

}

private void button3\_Click(object sender, EventArgs e)

{

}

}

}