Беслюбняк Віталій МІТ-21  
ПЗ№2

**ХІД РОБОТИ**

using System;

using System.Text;

using System.Security.Cryptography;

using System.Text.RegularExpressions;

namespace Lab2

{

class Globals

{

public static string? PlainText;

public static string? EncryptionKey;

public static string? Path;

}

class Keyboards

{

public static void Start()

{

Console.Clear();

Console.WriteLine(" [\*] Choose option [\*]");

Console.WriteLine("[=========================]");

Console.WriteLine(" | 1. Encrypt |");

Console.WriteLine(" | 2. Decrypt |");

Console.WriteLine(" | 3. Decoding |");

Console.WriteLine(" | 4. Exit |");

Console.WriteLine("[=========================]");

}

public static void Random()

{

Console.Clear();

Console.WriteLine(" [\*] Choose option [\*]");

Console.WriteLine("[=========================]");

Console.WriteLine(" | 1. Your key |");

Console.WriteLine(" | 2. Random key |");

Console.WriteLine(" | 3. Back |");

Console.WriteLine("[=========================]");

}

public static void File()

{

Console.Clear();

Console.WriteLine(" [\*] Choose option [\*]");

Console.WriteLine("[=========================]");

Console.WriteLine(" | 1. From file |");

Console.WriteLine(" | 2. Into console |");

Console.WriteLine(" | 3. Back |");

Console.WriteLine("[=========================]");

}

}

public class Cipher

{

public static string generate\_sec\_random\_number(int length)

{

var randomNumberGenerator = new RNGCryptoServiceProvider();

var randomNumber = new byte[length];

randomNumberGenerator.GetBytes(randomNumber);

return Convert.ToBase64String(randomNumber);

}

private static string KeyRepeater(string key, int length)

{

while (key.Length < length)

{

key += key;

}

return key.Substring(0, length);

}

private static string XOR(byte[] PlainText, byte[] EncryptionKey)

{

int TextLength = PlainText.Length;

string Result = string.Empty;

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

{

Result += ((char)(PlainText[i] ^ EncryptionKey[i]));

}

return Result;

}

public static void Encrypt(string? plainMessage, string? password, string? path, bool random = true, bool fileMode = false)

{

if (random == true)

{

if (fileMode == true)

{

try

{

plainMessage = File.ReadAllText(path);//"plaintext.txt"

byte[] ValidMessage = Encoding.UTF8.GetBytes(plainMessage);

byte[] ValidPassword = Encoding.UTF8.GetBytes(generate\_sec\_random\_number(plainMessage.Length));

string answer = XOR(ValidMessage, ValidPassword);

byte[] Result = Encoding.UTF8.GetBytes(answer);

File.WriteAllBytes("EncryptedFile.dat", Result);

Console.WriteLine("[#] File was encrypted! [#]");

Console.Write("\n[#] Encryption Key: \'{0}\' [#]", Encoding.UTF8.GetString(ValidPassword));

}

catch (System.Exception)

{

Console.WriteLine("[!] File not found [!]");

throw;

}

}

else

{

byte[] ValidMessage = Encoding.UTF8.GetBytes(plainMessage);

byte[] ValidPassword = Encoding.UTF8.GetBytes(generate\_sec\_random\_number(plainMessage.Length));

string answer = XOR(ValidMessage, ValidPassword);

Console.Write("\n[#] Encrypted text: \'{0}\' [#]", answer);

Console.Write("\n[#] Encryption Key: \'{0}\' [#]", Encoding.UTF8.GetString(ValidPassword));

}

}

else

{

if (fileMode == true)

{

try

{

plainMessage = File.ReadAllText(path);

byte[] ValidMessage = Encoding.UTF8.GetBytes(plainMessage);

byte[] ValidPassword = Encoding.UTF8.GetBytes(KeyRepeater(password, plainMessage.Length));

string answer = XOR(ValidMessage, ValidPassword);

byte[] Result = Encoding.UTF8.GetBytes(answer);

File.WriteAllBytes("ecr.dat", Result);

Console.WriteLine("[#] File was encrypted! [#]");

Console.Write("\n[#] Encryption Key: \'{0}\' [#]", Encoding.UTF8.GetString(ValidPassword));

}

catch (System.Exception)

{

throw;

}

}

else

{

byte[] ValidMessage = Encoding.UTF8.GetBytes(plainMessage);

byte[] ValidPassword = Encoding.UTF8.GetBytes(KeyRepeater(password, plainMessage.Length));

string answer = XOR(ValidMessage, ValidPassword);

Console.Write("\n[#] Encrypted text: \'{0}\' [#]", answer);

Console.Write("\n[#] Encryption Key: \'{0}\' [#]", Encoding.UTF8.GetString(ValidPassword));

}

}

}

public static string Decrypt(string? EncryptedMessage, string? password, string? path, bool fileMode = false)

{

if (fileMode == true)

{

try

{

EncryptedMessage = File.ReadAllText(path);//"entext.dat"

byte[] ValidMessage = Encoding.UTF8.GetBytes(EncryptedMessage);

int len = ValidMessage.Length;

byte[] ValidPassword = Encoding.UTF8.GetBytes(KeyRepeater(password, len));

string answer = XOR(ValidMessage, ValidPassword);

File.WriteAllText("decr.txt", answer);

// Console.Write("\n[#] Decrypted text: \'{0}\' [#]", answer);

// Console.Write("\n[#] Decryption Key: \'{0}\' [#]", Encoding.UTF8.GetString(ValidPassword));

// Console.WriteLine("[#] File was decrypted! [#]");

return answer;

}

catch (System.Exception)

{

Console.WriteLine("[!] File not found [!]");

throw;

}

}

else

{

byte[] ValidMessage = Encoding.UTF8.GetBytes(EncryptedMessage);

byte[] ValidPassword = Encoding.UTF8.GetBytes(KeyRepeater(password, EncryptedMessage.Length));

string answer = XOR(ValidMessage, ValidPassword);

Console.Write("\n[#] Decrypted text: \'{0}\' [#]", answer);

Console.Write("\n[#] Decryption Key: \'{0}\' [#]", Encoding.UTF8.GetString(ValidPassword));

return answer;

}

}

}

class Program

{

static void Main(string[] args)

{

do

{

Keyboards.Start();

string answer = Console.ReadLine();

switch (answer)

{

case "1":

Console.Write("\n");

Keyboards.File();

string answer\_case\_1 = Console.ReadLine();

switch (answer\_case\_1)

{

case "1":

Console.WriteLine("[\*] Enter file path [\*]");

Console.WriteLine("[!] File must be in one of the following formats: destination\_path/sample.txt; sample.txt (if in the same folder with program) [!]");

Globals.Path = Console.ReadLine();

Keyboards.Random();

string answer\_case\_2 = Console.ReadLine();

switch (answer\_case\_2)

{

case "1":

Console.WriteLine("[\*] Enter your key [\*]");

Globals.EncryptionKey = Console.ReadLine();

Cipher.Encrypt(path: Globals.Path, password: Globals.EncryptionKey, random: false, fileMode: true, plainMessage: null);

Console.ReadKey();

break;

case "2":

Cipher.Encrypt(path: Globals.Path, password: null, random: true, fileMode: true, plainMessage: null);

Console.ReadKey();

break;

case "3":

break;

}

break;

case "2":

Console.WriteLine("[\*] Enter plain text [\*]");

Globals.PlainText = Console.ReadLine();

Keyboards.Random();

string answer\_case\_3 = Console.ReadLine();

switch (answer\_case\_3)

{

case "1":

Console.WriteLine("[\*] Enter your password [\*]");

Globals.EncryptionKey = Console.ReadLine();

Cipher.Encrypt(path: null, password: Globals.EncryptionKey, random: false, fileMode: false, plainMessage: Globals.PlainText);

Console.ReadKey();

break;

case "2":

Cipher.Encrypt(path: null, password: null, random: true, fileMode: false, plainMessage: Globals.PlainText);

Console.ReadKey();

break;

case "3":

break;

}

break;

case "3":

break;

}

break;

case "2":

Console.Write("\n");

Keyboards.File();

string answer\_case\_4 = Console.ReadLine();

switch (answer\_case\_4)

{

case "1":

Console.WriteLine("[\*] Enter file path [\*]");

Console.WriteLine("[!] File must be in one of the following formats: destination\_path/sample.dat; sample.dat (if in the same folder with program) [!]");

Globals.Path = Console.ReadLine();

Console.WriteLine("[\*] Enter encryption key[\*]");

Globals.EncryptionKey = Console.ReadLine();

Cipher.Decrypt(EncryptedMessage: null, password: Globals.EncryptionKey, path: Globals.Path, fileMode: true);

Console.ReadKey();

break;

case "2":

Console.WriteLine("[\*] Enter plain text [\*]");

Globals.PlainText = Console.ReadLine();

Console.WriteLine("[\*] Enter encryption key [\*]");

Globals.EncryptionKey = Console.ReadLine();

Cipher.Decrypt(EncryptedMessage: Globals.PlainText, password: Globals.EncryptionKey, path: null, fileMode: false);

Console.ReadKey();

break;

case "3":

break;

}

break;

case "3":

string key = " ";

int am = key.Length;

Globals.Path = "enc.dat";

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

{

Globals.EncryptionKey = key.Insert(i, "Mit21");

Globals.EncryptionKey.Substring(0, am);

Console.WriteLine("\n Before key \'{0}\' ", Globals.EncryptionKey);

string checker = Cipher.Decrypt(EncryptedMessage: null, password: Globals.EncryptionKey, path: Globals.Path, fileMode: true);

Globals.EncryptionKey = checker.Substring(i, 5);

Console.WriteLine("\n Curr res {0}", checker);

Console.WriteLine("\n Curr key {0}", Globals.EncryptionKey);

Console.WriteLine("\n Curr index {0} \n", i);

string checker2 = Cipher.Decrypt(EncryptedMessage: null, password: Globals.EncryptionKey, path: Globals.Path, fileMode: true);

Console.ReadKey();

}

Console.ReadKey();

break;

case "4":

Console.Write("\n");

Environment.Exit(0);

break;

}

} while (true);

}

}

}