**Profesor coordonator: Elevi:**

**Iulia Lincan Popescu Catalina**

**Iancu Alina Bianca**

Tema proiectului:

Metode de codificare a mesajelor folosind algoritmi simpli sau chei de criptare alese de utilizator.

Motivarea alegerii temei:

Interesul in ceea ce priveste criptografia ne-a fost starnit inca de la inceputul descoperirii programarii si a componentei software a calculatorului, insa ceea ce ne-a determinat sa aprofundam aceasta stiinta a fost romanul cunoscutului autor de beletristica Dan Brown - “Digital Fortress”. Acesta fundamenteaza conceptia de criptare si prezinta metodele folosite, in special de agentia americana de securitate a informatiei NSA pentru a sparge anumite mesaje codate si a preveni astfel atacurile teroriste sau scurgerile de informatii.

Cerințe minime sistem:

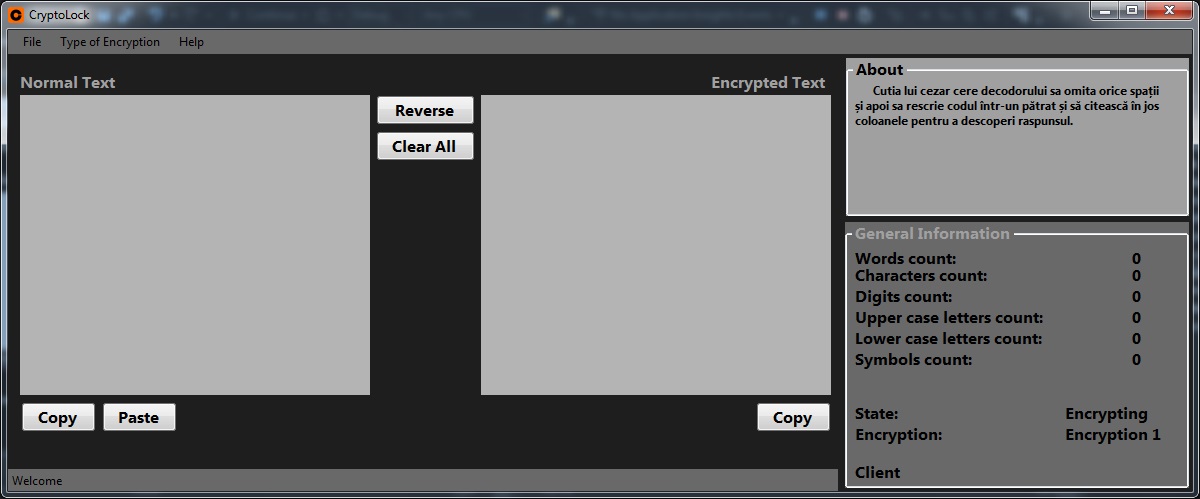
Programul este conceput sa foloseasca resursele regasite în varianta 2.0 a Microsoft .NET Framework. Intrucat sistemele de operare începand cu Windows Vista, 7, 8, 8.1 au pre-instalate versiunile 3.0, 3.5, 4.0 și 4.5, decât de la Windows XP in jos este necesara instalarea a .NET Framework-ului cu versiunea minima 2.0 . Rezolutia minima pentru rularea optima a programului este de 1024x768.

Unelte folosite:

Microsoft Visual Studio 2013 – C#

Adobe Photoshop CS6 *(pentru crearea logo-ului)*

Proiectul reprezinta un software prin care mesajele se pot cripta si decripta pe baza a doua tipuri de codificare. Unul dintre acestea utilizeaza o metoda de criptare prin cheie, aleasa de utilizator, iar cel de-al doilea, un algoritm predefinit, folosit in Antichitate de imparatul roman Cezar.



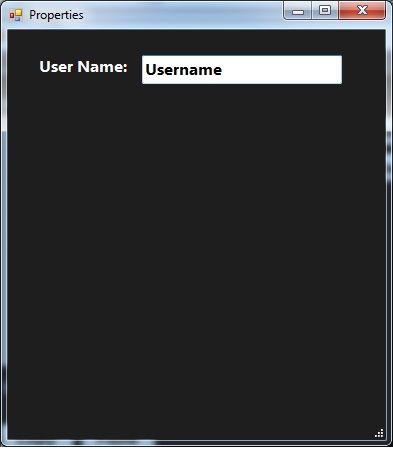
Initial am creat interfata vizuala a programului care consta in form-ul principal “Form1.cs”. Acesta contine doua casete text. Cea din stanga este folosita pentru introducerea textului ce se doreste a fi criptat/ decriptat, iar in caseta din dreapta se va afisa textul codificat/ decodificat. Cele doua casete comuta intre ele prin intermediul butonului “Reverse”, iar metodele de criptare se aleg din meniul “Type of Encryption”.

In continuare vom prezenta fiecare componenta a form-ului principal.

1. Meniul
2. File

* Properties

La accesarea acestei optiuni se deschide fereastra “Properties” ce permite schimbarea numelui de utilizator.



* Exit

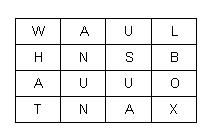
La accesarea optiunii “Exit” din meniul “File” se va parasi aplicatia.

1. Type of encryption

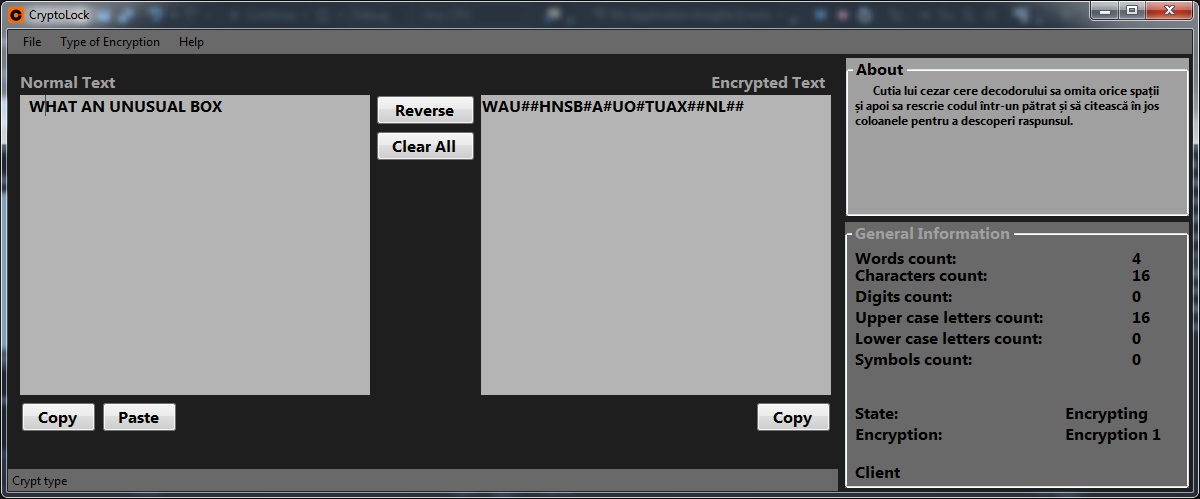
* Encryption 1

Acest tip de criptare utilizeaza un algoritm simplu de asezare a caracterelor intr-o matrice patratica, numit Cutia lui Cezar, completand-o cu caractere de tip “#” daca este necesar.

**Cutia lui Cezar** cere decodorului sa omita orice spații (noi le vom inlocui cu caractere diez) și apoi sa rescrie codul într-un pătrat și să citească în jos coloanele pentru a descoperi raspunsul, astfel încât, de exemplu, pentru a codifica expresia "WHAT AN UNUSUAL BOX” scriem caracterele într-un patrat, după cum urmează:



Codul criptat este : “WAU##HNSB#A#UO#TUAX##NL##” .



Codul pentru acest tip de criptare este urmatorul:

int x;

if (Math.Sqrt(richTextBox1.Text.Length) % 1 == 0)

x = (int)Math.Sqrt(richTextBox1.Text.Length);

else x = (int)Math.Sqrt(richTextBox1.Text.Length) + 1;

char[,] a = new char[x + 1, x + 1];

char[] c = new char[(int)Math.Pow(x, 2)];

char[] ex = new char[(int)Math.Pow(x, 2)];

for (int i = 0; i < c.Length; i++)

c[i] = '#';

for (int i = 0; i < richTextBox1.Text.Length; i++)

{

c[i] = richTextBox1.Text.ToCharArray()[i];

}

for (int i = 0; i < c.Length; i++)

{

if (c[i] == ' ') c[i] = '#';

}

int nr = -1;

for (int i = 1; i <= Math.Sqrt(c.Length); i++)

for (int j = 1; j <= Math.Sqrt(c.Length); j++) a[i, j] = c[++nr];

nr = -1;

for (int i = 1; i <= Math.Sqrt(c.Length); i++)

for (int j = 1; j <= Math.Sqrt(c.Length); j++) ex[++nr] = a[j, i];

string aux = new string(ex);

richTextBox2.Text = aux;

break;

* Encryption 2

Acest tip de criptare se bazeaza pe utilizarea unei chei. In criptografie, cheia reprezinta un parametru care determina rezultatul functional al unui algoritm criptografic sau al unui cifru. Aceasta cheie asigura securitate mesajului criptat, deoarece acesta poate fi spart doar prin cunoasterea cheii. O alta metoda de spargere a codului ar fi “forta bruta” care insa este foarte ineficienta ca timp, astfel la momentul decriptarii, mesajul nu va mai avea utilitate.

#### EXEMPLU:

Source = “hello world”  
Key = “hidden”

#### Impartirea in caractere:

Source:

h = 104 e = 101 l = 108 l = 108 o = 111 = 32 w = 119 o = 111 r = 114 l = 108 d = 100

Key:

h = 104 i = 105 d = 100 d = 100 e = 101 n = 110

#### Calcul:

Luam (source[n] ^ key[n1]) + 4 , unde n1 este o valoare rotativa (n1 < key.Length < n1). Acest lucru va amesteca cheia in sirul sursa. Singurele 3 metode de a le separa din nou sunt:

1. sa stii cheia pentru a ajunge la sursa
2. sa stii sursa ca sa ajungi la cheie
3. sa folosesti o metoda tip forta bruta ca sa ajungi la amandoua

(104 ^ 104) + 6 = 6 (101 ^ 105) + 6 = 18 (108 ^ 100) + 6 = 14 (108 ^ 100) + 6 = 14 (111 ^ 101) + 6 = 16 (32 ^ 110) + 6 = 84 (119 ^ 104) + 6 = 37 (111 ^ 105) + 6 = 12 (114 ^ 100) + 6 = 28 (108 ^ 100) + 6 = 14 (100 ^ 101) + 6 = 7

#### Conversia binara:

Dupa ce am efectuat operatiile precedente pentru sursa si cheie, transformam ce am obtinut in valori binare.

6 = 00000110 18 = 00010010 14 = 00001110 14 = 00001110 16 = 00010000 84 = 01010100 37 = 00100101 12 = 00001100 28 = 00011100 14 = 00001110 7 = 00000111

#### Transformarea intr-o secventa tip base 64:

#### Datele codate /criptate cu base64 ocupa aproximativ cu 33% mai mult spatiu decat datele originale, dar e util cand doriti sa transferati date sub forma de sir compact si criptat de la o aplicatie la alta (de exemplu de la partea de client /browser la un script pe server, si invers).

Prima data trebuie sa transformam valorile octetului binary intr-o secventa mare (un sir). Apoi separam acest sir in secvente de cate 4 caractere care se numesc nybbles (corespunde unei singure cifre hexazecimale sau unei jumatati de octet).

00000110000100100000111000001110000100000101 01000010010100001100000111000000111000000111

Separand aceasta in nybbles ar arata:

0000 0110 0001 0010 0000 1110 0000 1110 0001 0000 0101 0100 0010 0101 0000 1100 0001 1100 0000 1110 0000 0111

Si transformand inapoi in valori decimale:

0 6 1 2 0 14 0 14 1 0 5 4 2 5 0 12 1 12 0 14 0 7

Acum inmultim fiecare valoare cu 4 si adaugam m, unde m este o secventa de tipul 0,1,2,3.

0\*4+0, 6\*4+1, 1\*4+2, 2\*4+3, 0\*4+0, 14\*4+1, 0\*4+2, 14\*4+3, 1\*4+0, 0\*4+1, 5\*4+2, 4\*4+3, 2\*4+0, 5\*4+1, 0\*4+2, 12\*4+3, 1\*4+0, 12\*4+1, 0\*4+2, 14\*4+3, 0\*4+0, 7\*4+1

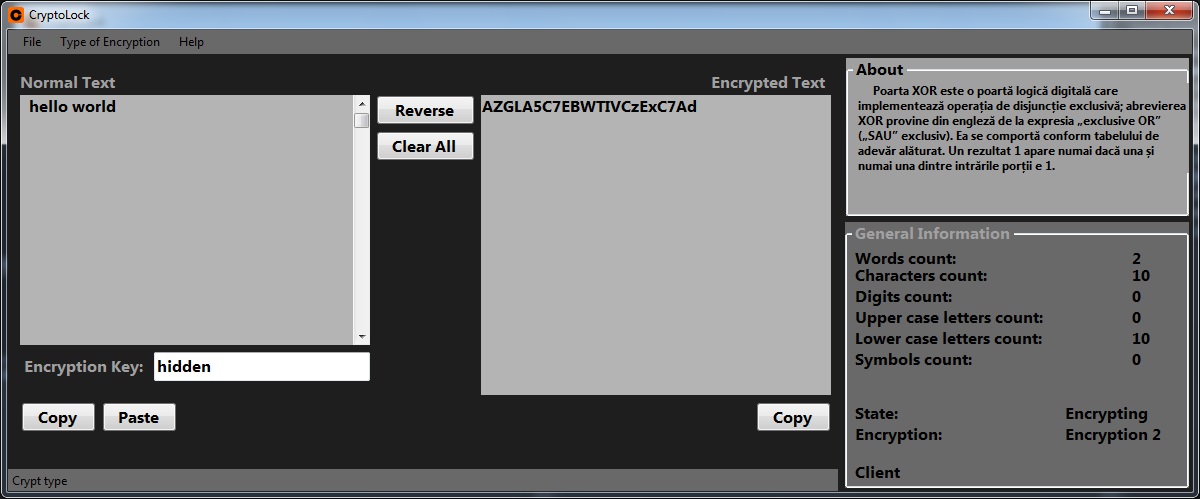
Aceste valori vor fi transformate in base64:

0 25 6 11 0 57 2 59 4 1 22 19 8 21 2 51 4 49 2 59 0 29 A Z G L A 5 C 7 E B W T I V C z E x C 7 A d

#### Codul de iesire:

Acesta este sirul codat in base64, care a fost creat din sursa: ”hello world” si cheia: “hidden”:

AZGLA5C7EBWTIVCzExC7Ad



Codul pentru acest tip de criptare este urmatorul:

using System;

using System.Collections.Generic;

using System.Text;

namespace XOREncryptionDecryption

{

static class XOREncryptionDecryption

{

private static string \_b64 = "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijk.mnopqrstuvwxyz-123456789+/=";

public static string Encode(string data, string key)

{

int keypos = 0;

string binarydata = "";

//convert the sting to a 8bit binary data

foreach (char c in data)

{

int xor = ((int)c ^ (int)key[keypos]) + (key.Length);

if (++keypos >= key.Length)

keypos = 0;

binarydata += DecToBinary(xor, 8);

}

int m = 0;

string cipher = "";

// splitt the binary string to 4 byte chunks and assign each chunk the proper b64 value

for (int i = 0; i < binarydata.Length; i += 4)

{

int v = BinToDec(binarydata.Substring(i, 4));

cipher += GetB64FromN(v \* 4 + m);

if (++m > 3)

m = 0;

}

return cipher;

}

public static string Decode(string data, string key)

{

int m = 0;

string binarydata = "";

// convert the b64 string to binary string

foreach (char c in data)

{

int v = (GetNFromB64(c) - m) / 4;

binarydata += DecToBinary(v, 4);

if (++m > 3)

m = 0;

}

// chop the 8bit binaries and mix back the key into it

int keypos = 0;

string decoded = "";

for (int i = 0; i < binarydata.Length; i += 8)

{

if (i + 8 > binarydata.Length)

break;

int c = BinToDec(binarydata.Substring(i, 8));

int dc = (c - key.Length) ^ (int)key[keypos];

if (++keypos >= key.Length)

keypos = 0;

decoded += new string((char)dc, 1);

}

return decoded;

}

// expects a base64 character and returns it's position in the base64 alphabet

private static int GetNFromB64(char n)

{

return \_b64.IndexOf(n);

}

// expects a position in the base64 alphabet and returns it's base64 character.

private static string GetB64FromN(int n)

{

if (n > \_b64.Length)

return "="; // well, we shouldn't reach this line. If we do, the encoding will be garbage anyway...

return new string(\_b64[n], 1);

}

// expects a integer value and length of the binary string (e.g. 4, 8, 16).

// returns the padded binary string

private static string DecToBinary(int value, int length)

{

// Declare a few variables we're going to need

string binString = "";

while (value > 0)

{

binString += value % 2;

value /= 2;

}

// we need to reverse the binary string

// that's why we are using this array stuff here.

string reverseString = "";

foreach (char c in binString)

reverseString = new string((char)c, 1) + reverseString;

binString = reverseString;

// do the padding here

binString = new string((char)'0', length - binString.Length) + binString;

return binString;

}

// expects the binary string and returns it's integer equivalent

private static int BinToDec(string Binary)

{

return Convert.ToInt32(Binary, 2);

}

}

}

1. Help

Accesand acest meniu se va deschide o fereastra ce ofera informatii privind modul in care aplicatia poate fi utilizata.

1. Comenzi de editare si casete text

* **Realizate prin controale de tip “Button”**

**Copy**- permite copierea textului introdus in caseta text

**Paste**- permite lipirea textului copiat anterior

**Reverse**- permite comutarea celor doua casete text

**Clear All**- permite stergerea textului din caseta curenta

* **Realizat prin control de tip “GroupBox”**

Caseta **General Information**- prezinta informatii generale privind aplicatia create, cum ar fi numele utilizatorului, tipul de criptare folosit, numarul de cuvinte sau caractere utilizate.

* **Realizat prin control de tip “Label”**

Fereastra **About**- ofera informatii privind metoda de criptare curenta.

**„Bazele Visual C#”**

editura Teora, Bucuresti, editia mai 1999

Traducere din limba engleza dupa editia originala,

„Essential VISUAL C#”,

publicata in Statele Unite ale Americii in anul 1995,

de catre editura Teora.

**„Programarea Orientatã pe Obiecte si**

**Programarea Vizualã cu C# .Net”**

Editata de Microsoft – M. Tataran, P. Jucovschi, T.I. Salomie in colaborarea cu N. Dumitriu-Lupan, R. Pintea, A. Nita, M. Nita, C. Sichim, N. Olaroiu

**Adrese web**

<http://msdn.microsoft.com/vstudio/express/visualc/easytolearn/>

http://en.wikipedia.org/wiki/XOR\_gate