Hill

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Hill

{

internal class Program

{

public class Hill

{

public int[,] keyMatrix { get; set; }

public int maTrixSize { get; set; }

public Hill(int[,] keyMatrix)

{

this.keyMatrix = keyMatrix;

this.maTrixSize = (int)Math.Sqrt(keyMatrix.Length);

}

public int Mod(int a, int b)

{

return (a % b + b) % b;

}

public int[] encript(int[] vector)

{

int[] encriptVector = new int[maTrixSize];

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

{

encriptVector[i] = 0;

for(int j = 0; j < maTrixSize; j++)

{

encriptVector[i]+= vector[j] \* keyMatrix[i, j];

}

encriptVector[i] = Mod(encriptVector[i], 26);

}

return encriptVector;

}

public string MessageEncript(string message)

{

message = message.Replace(" ", "");

int messageSize = message.Length;

int paddingSize = messageSize % maTrixSize == 0 ? 0 : Mod((maTrixSize - messageSize), maTrixSize);

string paddingMes = message.PadRight(paddingSize + messageSize, 'X');

string MesEncript = "";

int[] vector = new int[maTrixSize];

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

{

string msg = paddingMes.Substring(i, maTrixSize);

for(int j = 0;j<maTrixSize;j++)

{

char offset = char.IsUpper(msg[j]) ? 'A' : 'a';

vector[j] = (int)(paddingMes[i + j] - offset);

}

int[] encriptVector = encript(vector);

for(int j = 0; j < maTrixSize; j++)

{

char offset = char.IsUpper(msg[j]) ? 'A' : 'a';

MesEncript += (char)(encriptVector[j] + offset);

}

}

return MesEncript;

}

public int[] decript(int[] vector)

{

int[] decriptVector = new int[maTrixSize];

int detA = Mod((keyMatrix[0, 0] \* keyMatrix[1, 1] - keyMatrix[1, 0] \* keyMatrix[0, 1]), 26);

int detA\_1 = 0;

for(int i = 0;i < 26; i++)

{

if(Mod(i \* detA, 26) == 1)

{

detA\_1 = i;

break;

}

}

int[,] keyDecript = new int[maTrixSize, maTrixSize];

keyDecript[0, 0] = Mod((keyMatrix[1, 1] \* detA\_1), 26);

keyDecript[1, 1] = Mod((keyMatrix[0, 0] \* detA\_1), 26);

keyDecript[1, 0] = Mod((-keyMatrix[1, 0] \* detA\_1), 26);

keyDecript[0, 1] = Mod((-keyMatrix[0, 1] \* detA\_1), 26);

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

{

decriptVector[i] = 0;

for(int j = 0;j < maTrixSize; j++)

{

decriptVector[i] += keyDecript[i, j] \* vector[j];

}

decriptVector[i] = Mod(decriptVector[i], 26);

}

return decriptVector;

}

public string MessageDecript(string message)

{

message = message.Replace(" ", "");

int messageSize = message.Length;

string MesDecript = "";

int[] vector = new int[maTrixSize];

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

{

string msg = message.Substring(i, maTrixSize);

for (int j = 0; j < maTrixSize; j++)

{

char offset = char.IsUpper(msg[j]) ? 'A' : 'a';

vector[j] = (int)(message[i + j] - offset);

}

int[] decriptVector = decript(vector);

for (int j = 0; j < maTrixSize; j++)

{

char offset = char.IsUpper(msg[j]) ? 'A' : 'a';

MesDecript += (char)(decriptVector[j] + offset);

}

}

return MesDecript;

}

}

public static int chiadu(int a, int b)

{

return (a % b + b) % b;

}

static void Main(string[] args)

{

Random random= new Random();

int[,] matrix = null;

int i = 0;

string message = null;

string encript = null;

string decript = null;

Hill cypher = null;

do

{

do

{

Console.WriteLine("Nhap 1. Tao thong diep can khoa.");

Console.WriteLine("Nhap 2. tao khoa K.");

Console.WriteLine("Nhap 3. Ma hoa.");

Console.WriteLine("Nhap 4. Giai ma.");

Console.Write("Nhap lua chon: ");

i = int.Parse(Console.ReadLine());

} while (i < 1 || i > 4);

switch(i)

{

case 1:

Console.Write("Nhap thong diep can khoa: ");

message = Console.ReadLine();

break;

case 2:

matrix = new int[2, 2];

int dau = 0;

do

{

for (int z = 0; z < 2; z++)

{

for (int j = 0; j < 2; j++)

{

matrix[z, j] = random.Next(0, 26);

}

}

dau = 1;

if (chiadu(matrix[0, 0] \* matrix[1, 1] - matrix[1, 0] \* matrix[0, 1], 26) == 0)

{

Console.ForegroundColor = ConsoleColor.Red;

Console.WriteLine("Ma tran vua nhap khong hop le do khong có nghich dao. yeu cau nhap lai");

Console.ForegroundColor = ConsoleColor.White;

dau = 0;

}

} while (dau == 0);

Console.ForegroundColor = ConsoleColor.Green;

Console.WriteLine("Khoa vua tao la: ");

for (int z = 0; z < 2; z++)

{

for (int j = 0; j < 2; j++)

{

Console.Write($"{matrix[z, j]} ");

}

Console.WriteLine();

}

Console.ForegroundColor = ConsoleColor.White;

break;

case 3:

if(string.IsNullOrEmpty(message) || matrix == null)

{

Console.ForegroundColor = ConsoleColor.Red;

Console.WriteLine("Ma tran chua nhap hoac chua nhap thong diep can khôa. yeu cau nhap");

Console.ForegroundColor = ConsoleColor.White;

}

else

{

cypher = new Hill(matrix);

encript = cypher.MessageEncript(message);

Console.ForegroundColor = ConsoleColor.Green;

Console.WriteLine($"Ban ma: {encript}");

Console.ForegroundColor = ConsoleColor.White;

}

break;

case 4:

if (string.IsNullOrEmpty(encript))

{

Console.ForegroundColor = ConsoleColor.Red;

Console.WriteLine("Chua ma hoa nen khong giai ma duoc. Yeu cau ma hoa");

Console.ForegroundColor = ConsoleColor.White;

}

else

{

decript = cypher.MessageDecript(encript);

Console.ForegroundColor = ConsoleColor.Green;

Console.WriteLine($"Ban ro ban dau: {decript}");

Console.ForegroundColor = ConsoleColor.White;

}

break;

}

} while (i > 0 || i < 4);

Console.ReadKey();

}

}

}

Affine

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace AffineCipher

{

public class AffineCipher

{

public static int Mod(int a, int b)

{

return (a % b + b) % b;

}

// Hàm tìm nghịch đảo modulo

public static int ModInverse(int a, int m)

{

a = Mod(a, m);

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

{

if (Mod(i \* a, 26) == 1)

return i;

}

return -1;

}

public static int gcd(int a, int b)

{

int r = a % b;

if (r == 0)

return b;

else

return gcd(b, r);

}

// Hàm mã hóa Affine

public static string Encript(string massage , int a, int b)

{

string encriptMes = "";

foreach(char c in massage)

{

if(c != ' ')

{

char offset = char.IsUpper(c) ? 'A' : 'a';

int x = Mod(a \* (c - offset) + b, 26) ;

encriptMes += (char)(offset + x);

}

else

{

encriptMes += c;

}

}

return encriptMes;

}

// Hàm giải mã Affine

public static string Decript(string message , int a, int b)

{

int aInverse = ModInverse(a, 26);

string decriptMes = "";

foreach(char c in message)

{

if (c != ' ')

{

int offset = char.IsUpper(c) ? 'A' : 'a';

int x = Mod(aInverse \* ((c - offset) - b), 26);

decriptMes += (char)(offset + x);

}

else

{

decriptMes += c;

}

}

return decriptMes;

}

}

public class Program

{

public static void Main(string[] args)

{

Random random = new Random();

string plainText = null;

int a =0 , b=0, chkkey = 0;

string encriptMes = null;

string decriptMes;

int luachon; ;

do

{

do

{

Console.WriteLine("1. Nhap chuoi can ma hoa: ");

Console.WriteLine("2. Nhap khoa: ");

Console.WriteLine("3. Ma hoa: ");

Console.WriteLine("4. Giai ma: ");

Console.WriteLine("5. Thoat: ");

luachon = int.Parse(Console.ReadLine());

} while (luachon < 1 || luachon > 5);

switch (luachon)

{

case 1:

Console.Write("Nhap chuoi can ma hoa: ");

plainText= Console.ReadLine();

break;

case 2:

do

{

a = random.Next(0, 26);

b = random.Next(0, 26);

if((a >= 0 && a <=25) && (b >=0 && b <=25) && AffineCipher.gcd(a, 26) == 1)

{

chkkey = 1;

}

else

{

Console.WriteLine("Khoa khong hop le.Yeu cau nhap lai");

chkkey = 0;

}

}while(chkkey == 0);

Console.WriteLine($"a = {a}, b = {b}");

break;

case 3:

if(string.IsNullOrEmpty(plainText) || chkkey == 0)

{

Console.WriteLine("Ban chua nhap khoa hoac chuoi ma hoa. Yeu cau nhap khoa và chuoi ma hoa");

}

else

{

encriptMes = AffineCipher.Encript(plainText, a, b);

Console.WriteLine($"Ban ma: {encriptMes}");

}

break;

case 4:

if (string.IsNullOrEmpty(encriptMes) || chkkey == 0)

{

Console.WriteLine("Ban chua ma hoa. Yeu cau ma hoa");

}

else

{

decriptMes = AffineCipher.Decript(encriptMes, a, b);

Console.WriteLine($"Ban ro: {decriptMes}");

}

break;

};

}while (luachon != 5);

Console.ReadLine();

}

}

}

Des

using System;

using System.Security.Cryptography;

using System.Text;

namespace DESAlgorithm

{

class Program

{

static string EncryptDES(string message, string key)

{

byte[] keyBytes = Encoding.ASCII.GetBytes(key);

byte[] messageBytes = Encoding.ASCII.GetBytes(message);

using (DESCryptoServiceProvider des = new DESCryptoServiceProvider())

{

des.Key = keyBytes;

des.Mode = CipherMode.ECB; // Electronic Codebook mode

des.Padding = PaddingMode.PKCS7;

ICryptoTransform encryptor = des.CreateEncryptor();

byte[] encryptedBytes = encryptor.TransformFinalBlock(messageBytes, 0, messageBytes.Length);

string encryptedMessage = Convert.ToBase64String(encryptedBytes);

return encryptedMessage;

}

}

static string DecryptDES(string encryptedMessage, string key)

{

byte[] keyBytes = Encoding.ASCII.GetBytes(key);

byte[] encryptedBytes = Convert.FromBase64String(encryptedMessage);

using (DESCryptoServiceProvider des = new DESCryptoServiceProvider())

{

des.Key = keyBytes;

des.Mode = CipherMode.ECB;

des.Padding = PaddingMode.PKCS7;

ICryptoTransform decryptor = des.CreateDecryptor();

byte[] decryptedBytes = decryptor.TransformFinalBlock(encryptedBytes, 0, encryptedBytes.Length);

string decryptedMessage = Encoding.ASCII.GetString(decryptedBytes);

return decryptedMessage;

}

}

static void Main(string[] args)

{

string message = null, key = null, encryptedMessage = null, decryptedMessage = null;

bool isRunning = true;

while (isRunning)

{

Console.OutputEncoding = Encoding.UTF8;

Console.WriteLine("--------- Menu ---------");

Console.WriteLine("1. Nhap thông điệp");

Console.WriteLine("2. Nhap khoa");

Console.WriteLine("3. Ma hoa");

Console.WriteLine("4. Giai ma");

Console.WriteLine("5. Thoat");

Console.Write("Nhập lựa chọn: ");

int choice;

Console.WriteLine("Hãy chọn 1-5.");

Console.Write("Nhập lựa chọn của bạn: ");

choice = int.Parse(Console.ReadLine());

switch (choice)

{

case 1:

Console.Write("Nhập thông điệp cần mã hóa: ");

message = Console.ReadLine();

break;

case 2:

Console.Write("Nhập khóa K (8 ký tự): ");

key = Console.ReadLine();

break;

case 3:

encryptedMessage = EncryptDES(message, key);

Console.WriteLine("Thông điệp đã mã hóa: " + encryptedMessage);

Console.WriteLine();

break;

case 4:

decryptedMessage = DecryptDES(encryptedMessage, key);

Console.WriteLine("Bản mã đã giải mã: " + decryptedMessage);

Console.WriteLine();

break;

case 5:

isRunning = false;

break;

}

}

}

}

}

Aes

using System;

using System.Security.Cryptography;

using System.Text;

namespace AESAlgorithm

{

class Program

{

static string EncryptAES(string message, string key)

{

byte[] keyBytes = Encoding.UTF8.GetBytes(key);

byte[] messageBytes = Encoding.UTF8.GetBytes(message);

using (Aes aes = Aes.Create())

{

aes.Key = keyBytes;

aes.Mode = CipherMode.CBC; // Cipher Block Chaining mode

aes.Padding = PaddingMode.PKCS7;

ICryptoTransform encryptor = aes.CreateEncryptor(aes.Key, aes.IV);

byte[] encryptedBytes = encryptor.TransformFinalBlock(messageBytes, 0, messageBytes.Length);

string encryptedMessage = Convert.ToBase64String(encryptedBytes);

return encryptedMessage;

}

}