# ДОДАТОК А

## Лістинг програмного коду

### А1 Файл MainWindow.xaml

```
<Window x:Class="Secure_Connect.MainWindow"
      xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
      xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
      xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
      xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
      xmlns:local="clr-namespace:Secure_Connect"
      mc:Ignorable="d"
      Title="Secure Connect v. 1" Height="450" Width="800" WindowStartupLocation="CenterScreen" Closing="Window_Closing" MaxWidth="800" MaxHeight="450" ResizeMode="CanMinimize"
      >
   <Grid>
      <Grid.ColumnDefinitions>
         <ColumnDefinition Width="156*"/>
         <ColumnDefinition Width="356*"/>
         <ColumnDefinition Width="283*"/>
      </Grid.ColumnDefinitions>
      <TextBox x:Name="IPsrvbox" HorizontalAlignment="Left" Height="23" Margin="10,43,0,0" TextWrapping="Wrap" Text="127.0.0.1" VerticalAlignment="Top" Width="164" Grid.ColumnSpan="2"/>
      <Label Content="IP Адреса отримувача" HorizontalAlignment="Left" Margin="10,14,0,0" VerticalAlignment="Top" RenderTransformOrigin="-4.438,0.538"/>
      <Button Content="Прийняти" HorizontalAlignment="Left" Margin="33.333,46,0,0" VerticalAlignment="Top" Width="75" Click="Button_Click" Grid.Column="1"/>
      <TextBox x:Name="Msgsrv" HorizontalAlignment="Left" Margin="256.333,67,0,0" TextWrapping="Wrap" Text="" VerticalAlignment="Top" Height="342" Width="370" FontFamily="Yu Gothic UI" FontSize="14" Grid.ColumnSpan="2" Grid.Column="1"/>
      <TextBox x:Name="Msgcl" HorizontalAlignment="Left" Height="306" Margin="10,78,0,0" TextWrapping="Wrap" Text="" VerticalAlignment="Top" Width="388" FontFamily="Yu Gothic UI" FontSize="14" Grid.ColumnSpan="2"/>
      <Button Content="Відправити" HorizontalAlignment="Left" Margin="10,389,0,0" VerticalAlignment="Top" Width="75" Click="Button_Click_1"/>
      <Menu HorizontalAlignment="Left" Height="20" VerticalAlignment="Top" Width="792" Grid.ColumnSpan="3">
```

```
        <MenuItem Header="Дії з ключовими даними">
            <MenuItem x:Name="gen" Header="Згенерувати" Click="gen_Click" />
            <MenuItem x:Name="load" Header="Завантажити" Click="load_Click" />
            <MenuItem x:Name="deletekey" Header="Видалити кдючові дані з пам'яті" Click="deletekey_Click" />

        </MenuItem>

        <MenuItem Header="Ваша адресса"  x:Name ="yourip" Click="yourip_Click">
        </MenuItem>
      </Menu>

   </Grid>
</Window>
```

А2 Файл pass entr.xaml

```
<Window x:Class="Secure_Connect.pass_entr"
      xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
      xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
      xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
      xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
      xmlns:local="clr-namespace:Secure_Connect"
      mc:Ignorable="d"
      Title="Введення паролю" Height="200" Width="200" VerticalAlignment="Center" HorizontalAlignment="Center" WindowStartupLocation="CenterScreen" ResizeMode="NoResize">
   <Grid>
      <Button Content="OK" HorizontalAlignment="Left" Margin="59,103,0,0" VerticalAlignment="Top" Width="75" Click="Button_Click"/>
      <PasswordBox x:Name="pastxt" HorizontalAlignment="Left" Margin="10,50,0,0" VerticalAlignment="Top" Width="172"/>
      <TextBlock HorizontalAlignment="Left" Margin="10,10,0,0" TextWrapping="Wrap" Text="Будь ласка введіть пароль до НКІ" VerticalAlignment="Top" Width="172" Height="44" FontFamily="Yu Gothic UI" FontSize="14"/>

   </Grid>
</Window>
```

А3 файл pass entr.xaml.cs

```
using System.Windows;
```

```
namespace Secure_Connect
{
   public partial class pass_entr : Window
   {   public  string pass;
      public pass_entr()
      {
         InitializeComponent();
      }
      private void Button_Click(object sender, RoutedEventArgs e)
      {

         if (pastxt.Password.ToString() == "")
         {   this.DialogResult = false; return; }

            pass = pastxt.Password.ToString();
            this.DialogResult = true;
      }
   }
}
```

А4 Файл Kalyna.cs

```
using System;
using System.Text;

namespace Secure_Connect
{
   public class Kalyna
   {

            private readonly byte[] sBox = new byte[256] {
          0x63, 0x7c, 0x77, 0x7b, 0xf2, 0x6b, 0x6f, 0xc5, 0x30, 0x01, 0x67, 0x2b, 0xfe, 0xd7,
0xab, 0x76,
         0xca, 0x82, 0xc9, 0x7d, 0xfa, 0x59, 0x47, 0xf0, 0xad, 0xd4, 0xa2, 0xaf, 0x9c, 0xa4,
0x72, 0xc0,
         0xb7, 0xfd, 0x93, 0x26, 0x36, 0x3f, 0xf7, 0xcc, 0x34, 0xa5, 0xe5, 0xf1, 0x71, 0xd8,
0x31, 0x15,
         0x04, 0xc7, 0x23, 0xc3, 0x18, 0x96, 0x05, 0x9a, 0x07, 0x12, 0x80, 0xe2, 0xeb, 0x27,
0xb2, 0x75,
```

```
        0x09, 0x83, 0x2c, 0x1a, 0x1b, 0x6e, 0x5a, 0xa0, 0x52, 0x3b, 0xd6, 0xb3, 0x29, 0xe3,
0x2f, 0x84,
        0x53, 0xd1, 0x00, 0xed, 0x20, 0xfc, 0xb1, 0x5b, 0x6a, 0xcb, 0xbe, 0x39, 0x4a, 0x4c,
0x58, 0xcf,
        0xd0, 0xef, 0xaa, 0xfb, 0x43, 0x4d, 0x33, 0x85, 0x45, 0xf9, 0x02, 0x7f, 0x50, 0x3c,
0x9f, 0xa8,
        0x51, 0xa3, 0x40, 0x8f, 0x92, 0x9d, 0x38, 0xf5, 0xbc, 0xb6, 0xda, 0x21, 0x10, 0xff,
0xf3, 0xd2,
        0xcd, 0x0c, 0x13, 0xec, 0x5f, 0x97, 0x44, 0x17, 0xc4, 0xa7, 0x7e, 0x3d, 0x64, 0x5d,
0x19, 0x73,
        0x60, 0x81, 0x4f, 0xdc, 0x22, 0x2a, 0x90, 0x88, 0x46, 0xee, 0xb8, 0x14, 0xde, 0x5e,
0x0b, 0xdb,
        0xe0, 0x32, 0x3a, 0x0a, 0x49, 0x06, 0x24, 0x5c, 0xc2, 0xd3, 0xac, 0x62, 0x91, 0x95,
0xe4, 0x79,
        0xe7, 0xc8, 0x37, 0x6d, 0x8d, 0xd5, 0x4e, 0xa9, 0x6c, 0x56, 0xf4, 0xea, 0x65, 0x7a,
0xae, 0x08,
        0xba, 0x78, 0x25, 0x2e, 0x1c, 0xa6, 0xb4, 0xc6, 0xe8, 0xdd, 0x74, 0x1f, 0x4b, 0xbd,
0x8b, 0x8a,
        0x70, 0x3e, 0xb5, 0x66, 0x48, 0x03, 0xf6, 0x0e, 0x61, 0x35, 0x57, 0xb9, 0x86, 0xc1,
0x1d, 0x9e,
        0xe1, 0xf8, 0x98, 0x11, 0x69, 0xd9, 0x8e, 0x94, 0x9b, 0x1e, 0x87, 0xe9, 0xce, 0x55,
0x28, 0xdf,
        0x8c, 0xa1, 0x89, 0x0d, 0xbf, 0xe6, 0x42, 0x68, 0x41, 0x99, 0x2d, 0x0f, 0xb0, 0x54,
0xbb, 0x16 };

            private readonly byte[] inverseSBox = new byte[256] {

        0x52, 0x09, 0x6A, 0xD5, 0x30, 0x36, 0xA5, 0x38, 0xBF, 0x40, 0xA3, 0x9E, 0x81,
0xF3, 0xD7, 0xFB,
        0x7C, 0xE3, 0x39, 0x82, 0x9B, 0x2F, 0xFF, 0x87, 0x34, 0x8E, 0x43, 0x44, 0xC4,
0xDE, 0xE9, 0xCB,
        0x54, 0x7B, 0x94, 0x32, 0xA6, 0xC2, 0x23, 0x3D, 0xEE, 0x4C, 0x95, 0x0B, 0x42,
0xFA, 0xC3, 0x4E,
        0x08, 0x2E, 0xA1, 0x66, 0x28, 0xD9, 0x24, 0xB2, 0x76, 0x5B, 0xA2, 0x49, 0x6D,
0x8B, 0xD1, 0x25,
        0x72, 0xF8, 0xF6, 0x64, 0x86, 0x68, 0x98, 0x16, 0xD4, 0xA4, 0x5C, 0xCC, 0x5D,
0x65, 0xB6, 0x92,
        0x6C, 0x70, 0x48, 0x50, 0xFD, 0xED, 0xB9, 0xDA, 0x5E, 0x15, 0x46, 0x57, 0xA7,
0x8D, 0x9D, 0x84,
```

```
0x90, 0xD8, 0xAB, 0x00, 0x8C, 0xBC, 0xD3, 0x0A, 0xF7, 0xE4, 0x58, 0x05, 0xB8,
0xB3, 0x45, 0x06,
0xD0, 0x2C, 0x1E, 0x8F, 0xCA, 0x3F, 0x0F, 0x02, 0xC1, 0xAF, 0xBD, 0x03, 0x01,
0x13, 0x8A, 0x6B,
0x3A, 0x91, 0x11, 0x41, 0x4F, 0x67, 0xDC, 0xEA, 0x97, 0xF2, 0xCF, 0xCE, 0xF0,
0xB4, 0xE6, 0x73,
0x96, 0xAC, 0x74, 0x22, 0xE7, 0xAD, 0x35, 0x85, 0xE2, 0xF9, 0x37, 0xE8, 0x1C,
0x75, 0xDF, 0x6E,
0x47, 0xF1, 0x1A, 0x71, 0x1D, 0x29, 0xC5, 0x89, 0x6F, 0xB7, 0x62, 0x0E, 0xAA,
0x18, 0xBE, 0x1B,
0xFC, 0x56, 0x3E, 0x4B, 0xC6, 0xD2, 0x79, 0x20, 0x9A, 0xDB, 0xC0, 0xFE, 0x78,
0xCD, 0x5A, 0xF4,
0x1F, 0xDD, 0xA8, 0x33, 0x88, 0x07, 0xC7, 0x31, 0xB1, 0x12, 0x10, 0x59, 0x27,
0x80, 0xEC, 0x5F,
0x60, 0x51, 0x7F, 0xA9, 0x19, 0xB5, 0x4A, 0x0D, 0x2D, 0xE5, 0x7A, 0x9F, 0x93,
0xC9, 0x9C, 0xEF,
0xA0, 0xE0, 0x3B, 0x4D, 0xAE, 0x2A, 0xF5, 0xB0, 0xC8, 0xEB, 0xBB, 0x3C, 0x83,
0x53, 0x99, 0x61,
0x17, 0x2B, 0x04, 0x7E, 0xBA, 0x77, 0xD6, 0x26, 0xE1, 0x69, 0x14, 0x63, 0x55,
0x21, 0x0C, 0x7D };

    public string genkey()
    {
        Random rand = new Random();
        byte[] Key = new byte[64];

        rand.NextBytes(Key);

        return Encoding.Default.GetString(Key);
    }
    byte[] addkey(byte[] message, byte[] key)
    {
        for (int i = 0; i < 64; i++)
        {
            message[i] = (byte)(message[i] + key[i]);
        }
        return message;
    }
    byte[] subkey(byte[] message, byte[] key)
    {
```

```
    for (int i = 0; i < 64; i++)
    {
        message[i] = (byte)(message[i] - key[i]);
    }
    return message;
}
byte[] f_xor(byte[] message, byte[] key)
{

    for (int i = 0; i < 64; i++)
    {
        message[i] = (byte)(message[i] ^ key[i]);
    }
    return message;
}
byte[] sbox(byte[] message)
{
    for (int i = 0; i < 64; i++)
    {
        message[i] = sBox[message[i]];
    }
    return message;
}

byte[] shiftRows(byte[] message)
{
    byte[] buff = new byte[8];
    for (int r = 0; r < 8; r++)
    {
        for (int i = 0; i < 8; i++)
        {
            buff[i] = message[i + r * 8];
        }
        for (int l = 0; l < r; l++)
        {
            byte last = buff[7];
            Array.Copy(buff, 0, buff, 1, 7);
            buff[0] = last;

        }
```

```
        for (int i = 0; i < 8; i++)
        {
            message[i + r * 8] = buff[i];
        }
    }
    return message;
}
byte[] invshiftRows(byte[] message)
{
    byte[] buff = new byte[8];
    for (int r = 0; r < 8; r++)
    {
        for (int i = 0; i < 8; i++)
        {
            buff[i] = message[i + r * 8];
        }
        for (int l = 0; l < r; l++)
        {
            byte last = buff[0];
            Array.Copy(buff, 1, buff, 0, 7);
            buff[7] = last;

        }
        for (int i = 0; i < 8; i++)
        {
            message[i + r * 8] = buff[i];
        }
    }
    return message;

}
public static byte FFMul(byte a, byte b)
{
    byte aa = a, bb = b, r = 0, t;
    while (aa != 0)
    {
        if ((aa & 1) != 0)
            r = (byte)(r ^ bb);
        t = (byte)(bb & 0x80);
        bb = (byte)(bb << 1);
```

```
            if (t != 0)
                bb = (byte)(bb ^ 0x1b);
            aa = (byte)((aa & 0xff) >> 1);
        }
        return r;
    }
    private static byte[] InvMixColumns(byte[] s)
    {
        int[] sp = new int[4];
        byte b02 = (byte)0x0e, b03 = (byte)0x0b, b04 = (byte)0x0d, b05 = (byte)0x09;
        for (int j= 0; j < 16; j++)
        {
    int c = 0;
    sp[0] = FFMul(b02, s[0+c*4]) ^ FFMul(b03, s[1+c * 4]) ^ FFMul(b04, s[2+c * 4])
^ FFMul(b05, s[3+c * 4]);
            sp[1] = FFMul(b05, s[0 + c * 4]) ^ FFMul(b02, s[1 + c * 4]) ^ FFMul(b03, s[2
+ c * 4]) ^ FFMul(b04, s[3+c * 4]);
            sp[2] = FFMul(b04, s[0 + c * 4]) ^ FFMul(b05, s[1 + c * 4]) ^ FFMul(b02, s[2
+ c * 4]) ^ FFMul(b03, s[3 + c * 4]);
            sp[3] = FFMul(b03, s[0 + c * 4]) ^ FFMul(b04, s[1 + c * 4]) ^ FFMul(b05, s[2
+ c * 4]) ^ FFMul(b02, s[3 + c * 4]);

    for (int i = 0; i < 4; i++)
            s[i + c * 4] = (byte)(sp[i]);
        }

        return s;
    }

    private static byte[] MixColumns(byte[] s)
    {
        int[] sp = new int[4];
        byte b02 = (byte)0x02, b03 = (byte)0x03;
        for (int j = 0; j < 16; j++)
        {
    int c = 0;
    sp[0] = FFMul(b02, s[0 + c * 4]) ^ FFMul(b03, s[1 + c * 4]) ^ s[2 + c * 4] ^ s[3 + c
* 4];
            sp[1] = s[0 + c * 4] ^ FFMul(b02, s[1 + c * 4]) ^ FFMul(b03, s[2 + c * 4]) ^
s[3 + c * 4];
```

```
                sp[2] = s[0 + c * 4] ^ s[1 + c * 4] ^ FFMul(b02, s[2 + c * 4]) ^ FFMul(b03, s[3
+ c * 4]);
                sp[3] = FFMul(b03, s[0 + c * 4]) ^ s[1 + c * 4] ^ s[2 + c * 4] ^ FFMul(b02, s[3
+ c * 4]);
                for (int i = 0; i < 4; i++)
                    s[i + c * 4] = (byte)(sp[i]);
            }

            return s;
        }

        byte[] sboxinv(byte[] message)
        {
            for (int i = 0; i < 64; i++)
            {
                message[i] = inverseSBox[message[i]];
            }
            return message;
        }
        public void roundkey(ref byte[][] roundskey, byte[] key) {
            for(int i = 0; i < 18; i++)
            {
                roundskey[i] = key;
                for (int j = 0; j < 64; j++) {
                    roundskey[i][j] = (byte)((roundskey[i][j] >> 1) | ((roundskey[i][j] << (7)) &
255));
                }
            }
        }
        public string decr(string message, string key)
        {
      string result="";
            byte[] Keybyte = new byte[64];

            byte[][] roundsKey = new byte[18][];
            Array.Copy(Encoding.Default.GetBytes(key), Keybyte, 64);
           roundkey(ref roundsKey, Keybyte);
     int index = 0;

      while (index < Encoding.Default.GetBytes(message).Length)
```

```
    {
        byte[] messagebyte = new byte[64];
        int l = 0;
        if (index + 64 > message.Length) l = message.Length - index;
        else l = 64;

        Array.Copy(Encoding.Default.GetBytes(message), index, messagebyte, 0, l);

        for (int i = 0; i < 18; i++)
        {
            messagebyte = invshiftRows(messagebyte);
            messagebyte = sboxinv(messagebyte);
            messagebyte = subkey(messagebyte, roundsKey[i]);
            messagebyte = f_xor(messagebyte, roundsKey[i]);
            messagebyte = InvMixColumns(messagebyte);
        }

        result = result + Encoding.Default.GetString(messagebyte);
        index = index + 64;
    }
            return result;
        }

public string encr(string message, string key)
{
    string result = "";
    byte[] Keybyte = new byte[64];

            byte[][] roundsKey = new byte[18][];
            Array.Copy(Encoding.Default.GetBytes(key), Keybyte, 64);

            roundkey(ref roundsKey, Keybyte);
    int index = 0;
    while (index < Encoding.Default.GetBytes(message).Length)
    {  byte[] messagebyte = new byte[64];
        int l = 0;
        if (index+64>message.Length ) l = message.Length-index;
        else l = 64;

        Array.Copy(Encoding.Default.GetBytes(message), index, messagebyte, 0, l);
```

```
            for (int i = 0; i < 18; i++)
            {
               messagebyte = f_xor(messagebyte, Keybyte);
               messagebyte = addkey(messagebyte, Keybyte);
               messagebyte = sbox(messagebyte);
               messagebyte = shiftRows(messagebyte);
               messagebyte = MixColumns(messagebyte);
            }

            result = result + Encoding.Default.GetString(messagebyte);
            index = index + 64;
         }

            return result;
         }
   }
}
```

4А Файл MainWindow.xaml.cs

```
using System;
using System.Text;
using System.Windows;
using System.Threading;
using Microsoft.Win32;
using System.Security.Cryptography;
using System.IO;
using System.Net.Sockets;
using System.Net;

namespace Secure_Connect
{

   public partial class MainWindow : Window
   {
      string KEYForKalyna;
      int keyactive = 0;
      string IPsrv = "127.0.0.1";
      Kalyna Alg = new Kalyna();
```

```
public MainWindow()
{

   InitializeComponent();
   object locker = new object();
   Thread Server = new Thread(delegate () { Srv(); });

   Server.Start();


}

public void Srv()
{

   try
   {


      string txt;
      UdpClient server = new UdpClient(8111);

      IPEndPoint remoteIp = null;
      byte[] data;


      while (true)
      {
         if (keyactive == 1)
         {
            string ipcl = "";
            do
            {
               data = server.Receive(ref remoteIp);

            }

            while (server.Available > 0);

            ipcl = ipcl + remoteIp.ToString();
```

```
                ipcl = ipcl.Split(':')[0];

                txt = Encoding.Unicode.GetString(data);
                txt = Alg.decr(txt, KEYForKalyna);/// Розшифрування інформації
                string outtxt = DateTime.Now.ToShortTimeString() + " IP:" + ipcl + " - " + txt
+ "\n";
                Dispatcher.BeginInvoke(new ThreadStart(delegate { Msgsrv.Text += outtxt;
}));


            } Thread.Sleep(200);
          }
        }
        finally
        {
        }

    }


    public void Cl(string message, string IPsrv)
    {
        int port = 8111; // порт серверу
        UdpClient sender = new UdpClient();
        string msg;

        msg = Alg.encr(message, KEYForKalyna);///зашифрування інформації
        byte[] data = Encoding.Unicode.GetBytes(msg);
        sender.Send(data, data.Length, IPsrv, port);
    }
    private void Button_Click(object sender, RoutedEventArgs e)
    {
        IPsrv = IPsrvbox.Text;
        MessageBox.Show("Адреса отримувача " + IPsrv + " збережена!");
    }

    private void Button_Click_1(object sender, RoutedEventArgs e)
    {
        if (keyactive == 1)
        {
```

```
            string msg = Msgcl.Text;
            Thread Client = new Thread(delegate () { Cl(msg, IPsrv); });
            Client.Start();

        }
        else { MessageBox.Show("Перед початком роботи завантажте ключовий носій!");
}
    }

    private void load_Click(object sender, RoutedEventArgs e)
    {

        string keyfilepatch = "", keyfromfilehach = "";
        OpenFileDialog ofd = new OpenFileDialog();
        Nullable<bool> resultfile = ofd.ShowDialog();
        if (resultfile == true)
        {
            keyfilepatch = ofd.FileName;
        }
        else return;
        pass_entr ps = new pass_entr();

        Nullable<bool> resultpass = ps.ShowDialog();
        StreamReader sr = new StreamReader(keyfilepatch);
        UnicodeEncoding uniEncoding = new UnicodeEncoding();
        byte[] buff = new byte[512];
        int i = 0;
        while (sr.Peek() >= 0)
        {if (i > 512) break;
            buff[i] = (byte)sr.Read();
            i += 1;

        }

        string NKI;
        NKI = uniEncoding.GetString(buff);

        char[] buffer = new char[512];
        if (resultpass == true)
        {
```

```
    for (int x = 0, y = 0; x < NKI.Length; x++, y++)
    {
        if (y == ps.pass.Length)
            y = 0;
        buffer[x] = (char)(NKI[x] ^ ps.pass[y]);


    }
    NKI = new string(buffer);


    string[] NKId = new string[2];
       NKId[0]=NKI.Substring(0, 24);
        NKId[1] = NKI.Substring(24, 64);

    keyfromfilehach = NKId[0];
    sr.Close();


        var md5 = MD5.Create();
        var hashpass = md5.ComputeHash(Encoding.UTF8.GetBytes(ps.pass));
        string passwordhash = Convert.ToBase64String(hashpass);

        if (passwordhash == keyfromfilehach)
        {
            keyactive = 1;
            KEYForKalyna = NKId[1];///завантаження ключа з НКІ

            MessageBox.Show("Ключові данні завантажені до пам'яті програми ");

        }
        else { MessageBox.Show("Невірний пароль! "); }

    }
    else { MessageBox.Show("Ключові данні не завантажені до пам'яті програми"); }
}

private void deletekey_Click(object sender, RoutedEventArgs e)
{
    keyactive = 0;
```

```
    KEYForKalyna = "";
    MessageBox.Show("Ключові данні видалені з пам'яті програми");
}

private void gen_Click(object sender, RoutedEventArgs e)
{
    string password1 = "1";

    string password2 = "2";
    Stream wr;
    SaveFileDialog svdialog = new SaveFileDialog();
    svdialog.Filter = "txt files (*.txt)|*.txt";

    pass_entr ps1 = new pass_entr();
    Nullable<bool> resulps1 = ps1.ShowDialog();

    if (resulps1 == true)
    {
        password1 = ps1.pass;

        MessageBox.Show("Повторіть пароль");
        pass_entr ps2 = new pass_entr();
        Nullable<bool> resulps2 = ps2.ShowDialog();
        if (resulps2 == true)
        {
            password2 = ps2.pass;
            if (password1 != password2)
            {
                MessageBox.Show("Паролі не співпадають!");

                return;
            }
        }
    }


    if (svdialog.ShowDialog() == true)
    {

        if ((wr = svdialog.OpenFile()) != null)
```

```
            {
                string keykalyna = Alg.genkey();////Ключ що записуеться в НКІ

                UnicodeEncoding uniEncoding = new UnicodeEncoding();
                var md5 = MD5.Create();
                var hashpass = md5.ComputeHash(Encoding.UTF8.GetBytes(password1));
                string passwordhash = Convert.ToBase64String(hashpass);
                string NKI = "";
                char[] buffer = new char[88];
                NKI += passwordhash +  keykalyna;

                for (int i = 0, y = 0; i < NKI.Length; i++, y++)///шифрування паролем НКІ
                {
                    if (y == password1.Length)
                        y = 0;

                    buffer[i] = (char)(NKI[i] ^ password1[y]);
                }
                NKI = new string(buffer);

                wr.Write(uniEncoding.GetBytes(NKI), 0, uniEncoding.GetByteCount(NKI));

            }
            wr.Close();
        }
    }

    private void Window_Closing(object sender,
System.ComponentModel.CancelEventArgs e)
    {
        Environment.Exit(0);
    }

    private void yourip_Click(object sender, RoutedEventArgs e)
    {
        string host = Dns.GetHostName();
        IPAddress ip = Dns.GetHostByName(host).AddressList[0];
        MessageBox.Show("Ваша IP адресса " + ip);
    }
}}
```

| № док. | Позначення | Найменування | Дод. відом. |
|---|---|---|---|
| | | | |
| 1 | | Текстові документи | |
| | | | |
| | ГЮІК.ХХХХХ1412Ст.7ПЗ | Пояснювальна записка | 87 с. |
| | | | |
| 2 | | Графічні документи | |
| | | | |
| | | Презентаційний матеріал | _ А4. |
| | | | |
| 3 | | Інші документи | |
| | | | |
| | | Електронна версія ПЗ | 1 шт. |
| | | Відзив керівника | 1 с. |
| | | Рецензія | 1 с. |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |

| Изм. | Лист | № докум. | Підпис | Дата | ГЮІК. ХХХХХ1412Ст.7ВД | | |
|---|---|---|---|---|---|---|---|
| Розроб. | | Ковтун К.О. | | | Метод передачі повідомлень з використанням квантово-захищеного криптографічного алгоритму | Лит. | Лист 1 / Листів 1 |
| Перевір. | | Сєвєрінов О.В. | | | | Т | |
| | | | | | | ХНУРЕ Кафедра БІТ | |
| Н.контр. | | Конєва Н.Ф. | | | | | |
| Затверд. | | Халімов Г.З. | | | | | |