# Практическая работа №5

# Шифрование файлов

**Цели:**

1. Рассмотреть симметричный алгоритм шифрования RC2. 2.
2. Закрепить навыки использования криптографических классов платформы Microsoft .Net.

***Задание:*** Разработать приложение, которое позволит осуществлять шифрование и дешифрование файлов с использованием алгоритма RC2. Генерация ключа должна быть реализована из пароля, с использованием случайной примеси. Пароль должен иметь длину более восьми знаков и содержать строчные и прописные буквы, цифры, знаки математических операций. Случайная примесь должна храниться в начале файла и иметь длину восемь байт.

***Константы:*** отсутствуют.

***Переменные:*** KeyLen – Длина ключа(тип int)

## Исходные данные: Открытое сообщение «Открытое сообщение Пузанова Владимира Евгеньевича»

**Код MainForm:**

***Листинг 1 – Код MainForm:***

namespace RC4Application

{

public partial class MainForm : Form

{

// дескриптор криптопровайдера

IntPtr hProv;

// дескриптор случайного секретного ключа

IntPtr key;

// длина ключа

int KeyLength;

// буфер для массива байт шифруемой строки

byte[] toEncrypt;

// длина шифруемых и дешифруемых данных

uint PlainLen, CipherLen;

// буфер для массива дешифрованных байт

byte[] encrypted;

// определение функций, импортируемых из внешней библиотеки с реализацией криптопровайдера

// инициализация криптопровайдера

[DllImport("advapi32.dll", CharSet = CharSet.Auto, SetLastError = true)]

[return: MarshalAs(UnmanagedType.Bool)]

public static extern bool CryptAcquireContext(ref IntPtr hProv,

string pszContainer, string pszProvaider, uint dwProvType, uint dwFlags);

//структура PROV\_ENUMALGS\_EX для приема параметров криптопровайдера

[StructLayout(LayoutKind.Sequential)]

public struct PROV\_ENUMALGS\_EX

{

public int aiAlgid;

public int dwDefaultLen;

public int dwMinLen;

public int dwMaxLen;

public int dwProtocols;

public int dwNameLen;

[MarshalAs(UnmanagedType.ByValTStr, SizeConst = 20)]

public string szName;

public int dwLongNameLen;

[MarshalAs(UnmanagedType.ByValTStr, SizeConst = 40)]

public string szLongName;

}

// получение параметров криптопровайдера

[DllImport("advapi32.dll", CharSet = CharSet.Auto, SetLastError = true)]

[return: MarshalAs(UnmanagedType.Bool)]

public static extern bool CryptGetProvParam(IntPtr hProv, uint dwParam,

ref PROV\_ENUMALGS\_EX pbData, ref int pdwDataLen,

uint dwFlags);

// генерация случайного секретного ключа

[DllImport("advapi32.dll", CharSet = CharSet.Auto, SetLastError = true)]

[return: MarshalAs(UnmanagedType.Bool)]

public static extern bool CryptGenKey(IntPtr hProv, uint Algid,

uint dwFlags, ref IntPtr hKey);

// шифрование данных

[DllImport("advapi32.dll", CharSet = CharSet.Auto, SetLastError = true)]

[return: MarshalAs(UnmanagedType.Bool)]

public static extern bool CryptEncrypt(IntPtr hKey,

IntPtr hHash, int Final, uint dwFlags, byte[] pbData,

ref uint pdwDataLen, uint dwBufLen);

// дешифрование данных

[DllImport("advapi32.dll", CharSet = CharSet.Auto, SetLastError = true)]

[return: MarshalAs(UnmanagedType.Bool)]

public static extern bool CryptDecrypt(IntPtr hKey,

IntPtr hHash, int Final, uint dwFlags, byte[] pbData,

ref uint pdwDataLen);

// освобождение криптопровайдера

[DllImport("advapi32.dll", CharSet = CharSet.Auto, SetLastError = true)]

[return: MarshalAs(UnmanagedType.Bool)]

public static extern bool CryptReleaseContext(IntPtr hProv, uint dwFlags);

//результат выполнения функций, импортируемых из внешней библиотеки

bool res;

// массив возможных значений длины ключа

int[] ks;

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

{

// очистка редактора с шифротекстом

CipherText.Clear();

// декодирование введенной строки открытого текста

toEncrypt = Encoding.Unicode.GetBytes(PlainText.Text);

// полученние выбранной длины ключа

KeyLength = ks[KeyLen.SelectedIndex];

//задание длины секретного ключа для его генерации функцией CryptGenKey

KeyLength = KeyLength << 16;

// получение случайного секретного ключа для шифрования

res = CryptGenKey(hProv, 0x6801, (uint)KeyLength, ref key);

// если функция возвращает FALSE, то генерация исключения

if (!res)

throw new CryptographicException("Ошибка генерации ключа.");

// задание длины открытого текста

PlainLen = (uint)toEncrypt.Length;

//шифрование открытого текста (зашифрованные данные помещаются в буфер с открытым текстом)

res = CryptEncrypt(key, (IntPtr)null, 1, 0, toEncrypt, ref PlainLen,

(uint)toEncrypt.Length);

// если функция возвращает FALSE, то генерация исключения

if (!res)

throw new CryptographicException

("Ошибка шифрования данных.");

// получение шифротекста

encrypted = toEncrypt;

// отображение шифротекста

CipherText.Text = Encoding.Unicode.GetString(encrypted, 0,

(int)PlainLen);

// блокирование кнопки "Шифровать"

Encrypt.Enabled = false;

// блокирование списка для выбора длины ключа

KeyLen.Enabled = false;

// отчистка текстового поля с открытым сообщением

PlainText.Clear();

}

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

{

// отчистка строки с открытым текстом

PlainText.Clear();

// получение шифротекста

encrypted = Encoding.Unicode.GetBytes(CipherText.Text);

// создание буфера для дешифрованных данных

toEncrypt = new byte[encrypted.Length];

// задание длины дешифрованных данных

CipherLen = (uint)encrypted.Length;

//дешифрование данных (расшифрованные данные помещаются в буфер с шифротекстом)

res = CryptDecrypt(key, (IntPtr)null, 1, 0, encrypted, ref CipherLen);

// если функция возвращает FALSE, то генерация исключения

if (!res)

throw new CryptographicException

("Ошибка расшифрования данных.");

// отображение дешифрованного текста

toEncrypt = encrypted;

PlainText.Text = Encoding.Unicode.GetString(toEncrypt, 0,

toEncrypt.Length);

// блокирование кнопки "Дешифровать"

Decrypt.Enabled = false;

}

public MainForm()

{

InitializeComponent();

// инициализация криптопровайдера

if (!CryptAcquireContext(ref hProv, null, null, 1, 0))

//если первая попытка неудачна, инициализируем криптопровайдер с созданием контейнера ключей

if (!CryptAcquireContext(ref hProv, null, null, 1, 8))

throw new CryptographicException

("Ошибка при инициализации криптопровайдера.");

// получение параметров длины ключа для алгоритма RC4

// создание объекта для приема значений параметров

PROV\_ENUMALGS\_EX data = new PROV\_ENUMALGS\_EX();

// задание длины буфера для приема параметров

int dlen = 100;

//получение параметров для первого криптоалгоритма, реализованного в криптопровайдере

if (!CryptGetProvParam(hProv, 22, ref data, ref dlen, 1))

throw new CryptographicException

("Ошибка при получении параметров длины ключа.");

// цикл до получения параметров криптоалгоритма RC4

while (data.szName != "RC4")

CryptGetProvParam(hProv, 22, ref data, ref dlen, 2);

// получение минимально возможной длины ключа

KeyLength = data.dwMinLen;

// создание массива данных для списка

ks = new int[12];

// индексы элемента списка и выделенного элемента

int i = 0, isel = 0;

// максимальная длина ключа и длина ключа по умолчанию

int ksmax = data.dwMaxLen, ksdef = data.dwDefaultLen;

// получение шага изменения возможной длины ключа

// задание длины буфера дли приема данных

dlen = 4;

//первое поле объекта data в этом случае содержит шаг изменения длины ключа

if (!CryptGetProvParam(hProv, 35, ref data, ref dlen, 0))

throw new CryptographicException

("Ошибка при получении шага изменения длины ключа.");

// заполнение списка возможных значений длины ключа

do

{

ks[i] = KeyLength;

// сохранение индекса для длины ключа по умолчанию

if (KeyLength == ksdef)

isel = i;

// увеличение возможной длины ключа

KeyLength += data.aiAlgid;

i++;

}

while (KeyLength <= ksmax);

// связывание списка с массивом данных

KeyLen.DataSource = ks;

// выделение элемента списка

KeyLen.SelectedIndex = isel;

Encrypt.Enabled = PlainText.Text.Length == 0;

Decrypt.Enabled = CipherText.Text.Length == 0;

}

}

}}

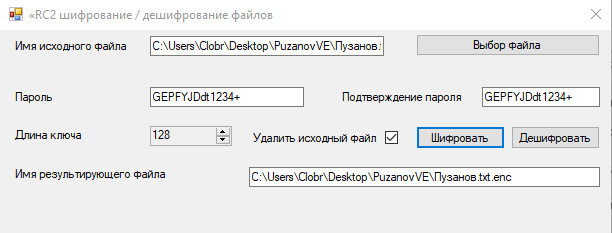


Рис.1 – шифрование файла

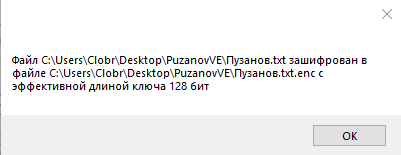


Рис.2 - Сообщение о результатах работы приложения

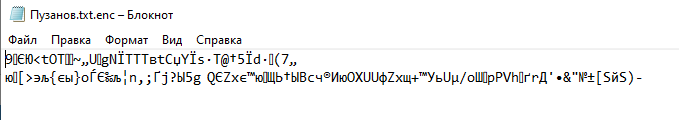


Рис.3 – Содержимое зашифрованного файла

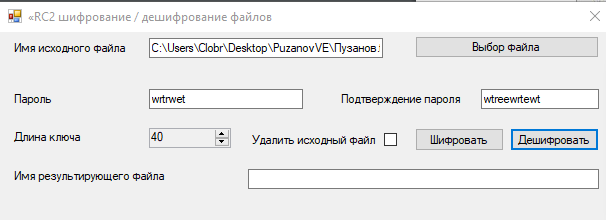


Рис.4 – Указание ложного пароля при дешифровании

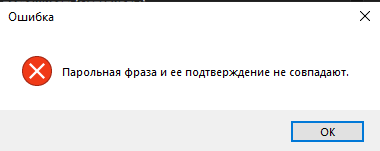


Рис.5 – Ошибка при указании ложного пароля при дешифровании

Выполнил студент Пузанов В. Е., ФИТУ 010304-КМСб-о22 Проверил ст. преподаватель каф. ПМ Черноиван Д.Н.