**Міністерство Освіти І НАУКИ України**

**Національний університет "Львівська політехніка"**

Інститут **КНІТ**

Кафедра **ПЗ**

### ЗВІТ

До лабораторної роботи № 3

**З дисципліни:** *“Безпека програм та даних”*

**На тему:** *“Створення програмного засобу для забезпечення*

*конфіденційності інформації”*

**Лектор:**

доц. каф. ПЗ

Сенів М. М.

**Виконав:**

ст. гр. ПЗ-44

Вонс Ю. А.

**Прийняв:**

асист. каф. ПЗ

Самбір А. А.

« \_\_\_\_ » \_\_\_\_\_\_\_\_ 2019 р.

∑= \_\_\_\_\_ .

Львів – 2019

**Тема роботи:** Створення програмного засобу для забезпечення конфіденційності інформації.

**Мета роботи:** Ознайомитись з методами криптографічного забезпечення конфіденційності інформації, навчитись створювати комплексні програмні продукти для захисту інформації з використанням алгоритмів симетричного шифрування, хешування та генераторів псевдовипадкових чисел.

**TЕОРЕТИЧНІ ВІДОМОСТІ**

  RC5 – це алгоритм симетричного шифрування, розроблений Роном Райвестом в середині 90-х років. При розробці RC5 ставилась задача досягнення наступних характеристик.

* Придатність для апаратної та програмної реалізації. Швидкість виконання.
* RC5 є простим алгоритмом, що працює з даними розміром в машинне слово.
* Адаптованість до процесорів з різною довжиною слова.
* Змінна кількість раундів.
* Змінна довжина ключа.
* Низькі вимоги до пам'яті.
* Високий ступінь захисту.
* Залежність циклічних зсувів від даних.

RC5 фактично являє собою родину алгоритмів шифрування, що визначається трьома наступними параметрами:

* Розмір слова в бітах (16, 32, 64);
* Кількість раундів (0 - 255);
* Кількість 8-бітових байтів (октетів) в таємному ключі К (0 - 255).

**ЗАВДАННЯ**

Згідно до варіанту, наведеного в таблиці, створити прикладну програму для шифрування інформації за алгоритмом RC5.

Програма повинна отримувати від користувача парольну фразу і, на її основі, шифрувати файли довільного розміру, а результат зберігати у вигляді файлу з можливістю подальшого дешифрування (при введенні тієї самої парольної фрази).

**ТЕКСТ ПРОГРАМИ**

public class RC5

{

public void Encrypt(byte[] key, Stream input, Stream output)

{

var s = GetSubkeys(key, 8);

var random = new Random(32, 17751, 0, (uint)DateTime.Now.ToBinary());

ulong a, b, a0 = 0, b0 = 0;

a = random.NextValue;

b = random.NextValue;

var inBuffer = new byte[16];

input.Position = 0;

output.Position = 0;

BlockEncoder(ref a0, ref b0, a, b, s, 8, output);

int readed = -1;

while (input.Position < input.Length)

{

readed = input.Read(inBuffer, 0, 16);

if (readed < 16)

{

ZeroFiller(inBuffer, readed, 16 - readed);

Buffer.BlockCopy(BitConverter.GetBytes(16 - readed), 0, inBuffer, 16 - 1, 1);

}

a = BitConverter.ToUInt64(inBuffer, 0);

b = BitConverter.ToUInt64(inBuffer, 8);

BlockEncoder(ref a0, ref b0, a, b, s, 8, output);

}

if (readed == 16)

{

ZeroFiller(inBuffer, 0, 16);

Buffer.BlockCopy(BitConverter.GetBytes(16), 0, inBuffer, 16 - 1, 1);

a = BitConverter.ToUInt64(inBuffer, 0);

b = BitConverter.ToUInt64(inBuffer, 8);

BlockEncoder(ref a0, ref b0, a, b, s, 8, output);

}

}

public void Decrypt(byte[] key, Stream input, Stream output)

{

var s = GetSubkeys(key, 8);

ulong A = 0, B = 0, a0 = 0, b0 = 0;

var inBuffer = new byte[16];

input.Position = 0;

output.Position = 0;

input.Read(inBuffer, 0, 16);

A = BitConverter.ToUInt64(inBuffer, 0);

B = BitConverter.ToUInt64(inBuffer, 8);

BlockDecoder(ref a0, ref b0, ref A, ref B, s, 8, output, false);

while (input.Position < input.Length - 16)

{

input.Read(inBuffer, 0, 16);

A = BitConverter.ToUInt64(inBuffer, 0);

B = BitConverter.ToUInt64(inBuffer, 8);

BlockDecoder(ref a0, ref b0, ref A, ref B, s, 8, output, true);

}

input.Read(inBuffer, 0, 16);

A = BitConverter.ToUInt64(inBuffer, 0);

B = BitConverter.ToUInt64(inBuffer, 8);

BlockDecoder(ref a0, ref b0, ref A, ref B, s, 8, output, false);

byte appendixLength = (byte)(B >> ((8 - 1) \* 8));

if (appendixLength < 16)

{

Buffer.BlockCopy(BitConverter.GetBytes(A), 0, inBuffer, 0, 8);

Buffer.BlockCopy(BitConverter.GetBytes(B), 0, inBuffer, 8, 8);

output.Write(inBuffer, 0, 16 - appendixLength);

}

}

protected static void ZeroFiller(Array a, int offset, int count)

{

for (var i = 0; i < count; i++) {

(a as dynamic)[offset + i] = (byte)0;

}

}

private static void BlockEncoder(ref ulong a0, ref ulong b0, ulong a, ulong b, ulong[] s, byte r,

Stream output)

{

a ^= a0;

b ^= b0;

a += s[0];

b += s[1];

for (int i = 1; i < r + 1; i++) {

a = (OffsetLeft((a ^ b), (int) b) + s[2 \* i]);

b = (OffsetLeft((b ^ a), (int) a) + s[2 \* i + 1]);

}

a0 = a;

b0 = b;

output.Write(BitConverter.GetBytes(a), 0, 8);

output.Write(BitConverter.GetBytes(b), 0, 8);

}

private static void BlockDecoder(ref ulong a0, ref ulong b0, ref ulong a, ref ulong b, ulong[] s, byte r,

Stream output, bool writeFlag)

{

ulong tmpA = a;

ulong tmpB = b;

for (byte i = r; i > 0; i--) {

b = (OffsetRight((b - s[2 \* i + 1]), (int) a) ^ a);

a = ((OffsetRight((a - s[2 \* i]), (int) b) ^ b));

}

b -= s[1];

a -= s[0];

a ^= a0;

b ^= b0;

a0 = tmpA;

b0 = tmpB;

if (writeFlag) {

output.Write(BitConverter.GetBytes(a), 0, 8);

output.Write(BitConverter.GetBytes(b), 0, 8);

}

}

private static ulong[] GetSubkeys(byte[] key, byte r)

{

const ulong P64 = 0xb7e151628aed2a6b;

const ulong Q64 = 0x9e3779b97f4a7c15;

ulong x = 0, y = 0, i, j;

var b = key.Length;

ulong c = (ulong)(b / 2);

var l = new ulong[c];

Buffer.BlockCopy(key, 0, l, 0, b);

ulong t = (ushort)(2 \* (r + 1));

var s = new ulong[t];

s[0] = P64;

for (i = 1; i < t; i++)

s[i] = (s[i - 1] + Q64);

i = j = 0;

for (ulong k = 0; k < (ulong) (2 \* r + 3); k++) {

x = s[i] = OffsetLeft((s[i] + x + y), 3);

y = l[j] = OffsetLeft((l[j] + x + y), (int)(x + y));

i = ((i + 1) % t);

j = ((j + 1) % c);

}

return s;

}

public static ulong OffsetLeft(ulong a, int offset)

{

offset %= 64;

return ((a << offset) | (a >> (64 - offset)));

}

public static ulong OffsetRight(ulong a, int offset)

{

offset %= 64;

return ((a >> offset) | (a << (64 - offset)));

}

}

**РЕЗУЛЬТАТИ**

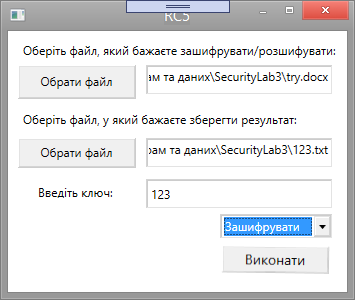


Рис. 1. Шифрування файлу

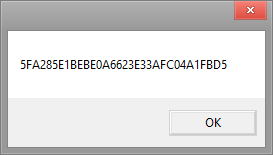


Рис. 2. Захешований ключ MD5-им

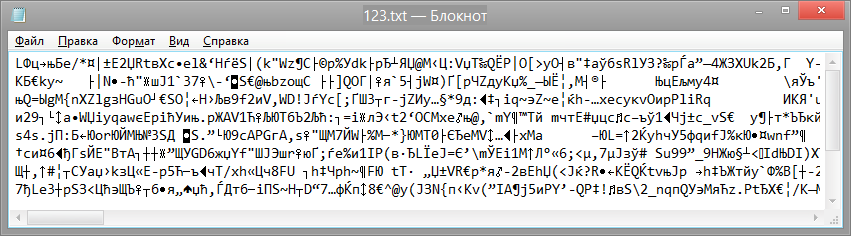


Рис. 3. Зашифрований файл

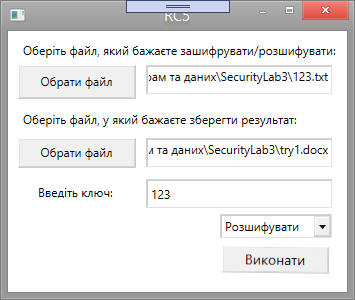


Рис. 4. Розшифрування файлу

**ВИСНОВКИ**

Виконуючи дану лабораторну роботу, я ознайомився з методами криптографічного забезпечення конфіденційності та розробив програмне забезпечення, що шифрує дані у файлі за допомогою симетричного алгоритму шифрування RC5.