# КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ імені ТАРАСА ШЕВЧЕНКА



## ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

**Кафедра прикладних інформаційних систем**

**Звіт до лабораторної роботи №10**

# з курсу

**«Безпека мереж і комп’ютерних систем»**

*Студента 2 курсу*

*групи ПП-21 спеціальності 122 «Комп'ютерні науки» ОП «Прикладне програмування»*

%username%

*Викладач:*

проф. Сайко В.Г.

## Київ – 202

**1.Назва роботи**

Потокові симетричні алгоритми шифрування.

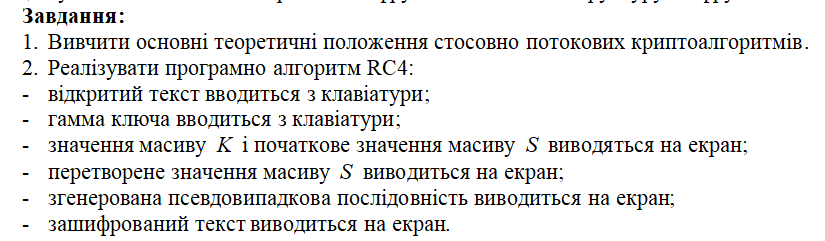
1. **Тема роботи**

Потокові симетричні алгоритми шифрування.

1. **Мета роботи**

Ознайомитися з основними поняттями, присвяченими принципам функціонування потокових алгоритмів шифрування. Вивчити структуру шифру RC4.

1. **Умова завдання**



1. **Рішення**

Код на С#:

using System;

using System.Text;

namespace lab10\_rc4\_

{

class Program

{

public static class RC4

{

public static byte[] Apply(byte[] data, byte[] key)

{

int[] S = new int[256];

for (int \_ = 0; \_ < 256; \_++)

{

S[\_] = \_;

}

int[] T = new int[256];

if (key.Length == 256)

{

Buffer.BlockCopy(key, 0, T, 0, key.Length);

}

else

{

for (int \_ = 0; \_ < 256; \_++)

{

T[\_] = key[\_ % key.Length];

}

}

int i = 0;

int j = 0;

for (i = 0; i < 256; i++)

{

j = (j + S[i] + T[i]) % 256;

int temp = S[i];

S[i] = S[j];

S[j] = temp;

}

i = j = 0;

byte[] result = new byte[data.Length];

for (int iteration = 0; iteration < data.Length; iteration++)

{

i = (i + 1) % 256;

j = (j + S[i]) % 256;

int temp = S[i];

S[i] = S[j];

S[j] = temp;

int K = S[(S[i] + S[j]) % 256];

result[iteration] = Convert.ToByte(data[iteration] ^ K);

}

return result;

}

}

static void Main(string[] args)

{

string phrase = "Lab Ten Example";

string key\_phrase = "Keep it secret. Keep it safe.";

byte[] data = Encoding.UTF8.GetBytes(phrase);

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

byte[] encrypted\_data = RC4.Apply(data, key);

byte[] decrypted\_data = RC4.Apply(encrypted\_data, key);

string decrypted\_phrase = Encoding.UTF8.GetString(decrypted\_data);

Console.WriteLine("Phrase:\t\t\t{0}", phrase);

Console.WriteLine("Phrase Bytes:\t\t{0}", BitConverter.ToString(data));

Console.WriteLine("Key Phrase:\t\t{0}", key\_phrase);

Console.WriteLine("Key Bytes:\t\t{0}", BitConverter.ToString(key));

Console.WriteLine("Encryption Result:\t{0}", BitConverter.ToString(encrypted\_data));

Console.WriteLine("Decryption Result:\t{0}", BitConverter.ToString(decrypted\_data));

Console.WriteLine("Decrypted Phrase:\t{0}", decrypted\_phrase);

Console.WriteLine(Environment.NewLine + "Press enter to close");

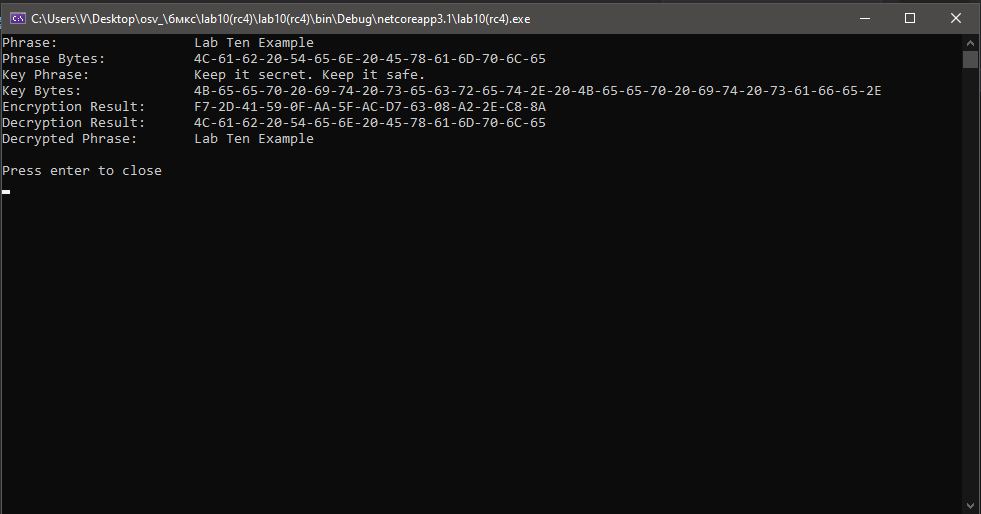
Console.ReadLine();

}

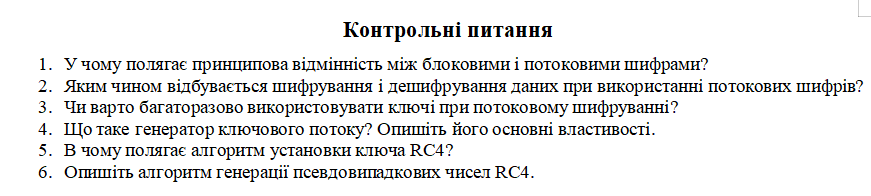
}

}

Результат роботи:



**Контрольні запитання:**



1. Потоковими називаються шифри, в яких потік цифрових даних шифрується послідовно біт за бітом або байт за байтом. Особливістю блочного шифру є обробка блоку декількох байт за одну ітерацію (як правило 8 або 16).
2. Випадковий потік бітів генерується по короткому секретному ключу за допомогою відкритого алгоритму, що називається генератором ключового потоку. Тут біти шифротекста розраховуються за правилом:

, 

де  – біти відкритого тексту;

 – біти ключового потоку.

Дешифрування відповідно описується співвідношенням:

.

1. Припустимо, що повідомлення ** та **були зашифровані одним ключем . Тоді зловмисник, перехопивши шифротексти, легко знайде суму по модулю 2 відкритих текстів :

.

Отже, необхідно міняти ключі або з кожним новим повідомленням, або з черговим сеансом зв'язку.

1. Щоб надати необхідну стійкість шифру, генератор ключового потоку виробляє рядок бітів з певними властивостями. Ключовий потік повинен мати:

*- великий період*;

*- псевдо-випадкові властивості*;

*- лінійну складність.*

1. Секретний ключ задається набором чисел, які поміщаються в ключовий масив , що також містить 255 елементів. Зазвичай вибирають коротку послідовність чисел, яка потім повторюється до заповнення .
2. Генерують байти псевдовипадкового ключового потоку, вибираючи випадкові елементи масиву  і змінюючи  для наступної вибірки:
3. параметрам  присвоюється нульове значення;
4. для генерації кожного байта випадкового потоку використовується наступний алгоритм :

* 
* 
*  і  міняються місцями;
* 
* 

**Висновки**

В результаті виконання даної лабораторної роботи я ознайомився з основними поняттями, присвяченими принципам функціонування потокових алгоритмів шифрування, вивчити структуру шифру RC4. Створив програмний засіб з реалізацією шифрувального алгоритму RC4, використовуючи мову програмування С#. Створив клас з методами, викликав їх у функції main, отримав та оцінив результат.

Вважаю дану лабораторну роботу виконаною в повному обсязі.