

**КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
імені ТАРАСА ШЕВЧЕНКА**



ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Кафедра прикладних інформаційних систем

Звіт до лабораторної роботи №9

3 курсу

«Безпека мереж і комп'ютерних систем»

*студента 2 курсу
групи ПП-22
спеціальності 122 «Комп'ютерні науки»
ОП «Прикладне програмування»
Шевлюк Вікторії Віталіївни*

*Перевірів:
д.т.н, професор
Сайко В. Г.*

Київ 2022

Тема: Поточкові симетричні алгоритми шифрування.

Мета: Ознайомитися з основними поняттями, присвяченими принципам функціонування поточкових алгоритмів шифрування. Вивчити структуру шифру RC4.

Завдання:

1. Вивчити основні теоретичні положення стосовно поточкових криптоалгоритмів.

2. Реалізувати програмно алгоритм RC4:

- ▶ відкритий текст вводиться з клавіатури;
- ▶ гамма ключа вводиться з клавіатури;
- ▶ значення масиву K і початкове значення масиву S виводяться на екран;
- ▶ перетворене значення масиву S виводиться на екран;
- ▶ згенерована псевдовипадкова послідовність виводиться на екран;
- ▶ зашифрований текст виводиться на екран.

Хід роботи:

Створимо мовою C# програму, що буде використовувати алгоритм RC4, при цьому відкритий текст вводиться з клавіатури; гамма ключа вводиться з клавіатури; значення масиву K і початкове значення масиву S виводяться на екран; перетворене значення масиву S виводиться на екран; згенерована псевдовипадкова послідовність виводиться на екран; зашифрований текст виводиться на екран.

Нижче додаю код програми:

```

using System;
using System.Text;

namespace lab9
{
    Ссылка: 0
    class Program
    {
        Ссылка: 2
        public static class RC4
        {
            Ссылка: 2
            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;
    }
}

```

Ссылка: 0

```

static void Main(string[] args)
{
    string phrase = "Viktoria Shevliuk PP22";

    string key_phrase = "What do you eat for breakfast?";

    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);

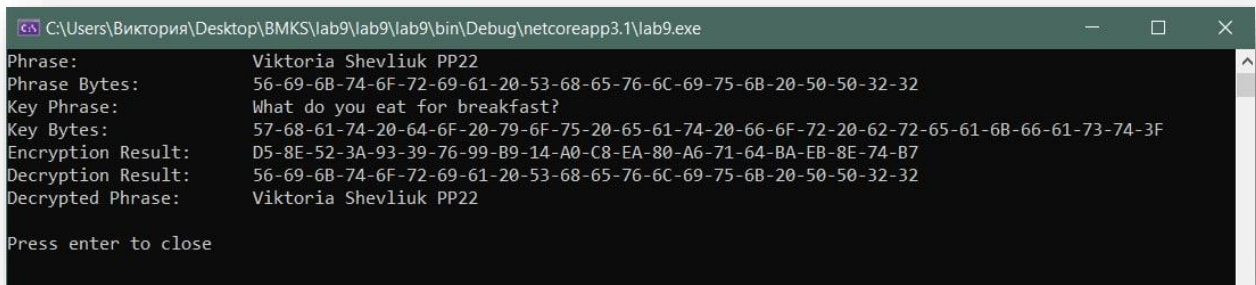
    string decrypted_phrase = Encoding.UTF8.GetString(decrypted_data);

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



```
C:\Users\Виктория\Desktop\BMKS\lab9\lab9\bin\Debug\netcoreapp3.1\lab9.exe
Phrase: Viktoria Shevliuk PP22
Phrase Bytes: 56-69-6B-74-6F-72-69-61-20-53-68-65-76-6C-69-75-6B-20-50-50-32-32
Key Phrase: What do you eat for breakfast?
Key Bytes: 57-68-61-74-20-64-6F-20-79-6F-75-20-65-61-74-20-66-6F-72-20-62-72-65-61-6B-66-61-73-74-3F
Encryption Result: D5-8E-52-3A-93-39-76-99-B9-14-A0-C8-EA-80-A6-71-64-BA-EB-8E-74-B7
Decryption Result: 56-69-6B-74-6F-72-69-61-20-53-68-65-76-6C-69-75-6B-20-50-50-32-32
Decrypted Phrase: Viktoria Shevliuk PP22
Press enter to close
```

Висновок: під час цієї лабораторної роботи я ознайомилася з основними поняттями, присвяченими принципам функціонування поточкових алгоритмів шифрування та вивчила структуру шифру RC4.

RC4 — група симетричних шифрів, які шифрують кожен символ відкритого тексту незалежно від інших символів., розроблена Роном Рівестом у 1987 році.

Потоковий шифр породжує послідовність елементів потоку ключа базуючись на внутрішньому стані. Цей стан оновлюється двома способами: якщо стан змінюється незалежно від відкритого тексту або шифротексту повідомлення, шифр позначають *синхронним* поточковим шифром. Натомість, поточкові шифри такі, що *самосинхронізуються* оновлюють стан на основі попередніх цифр шифротексту.

Контрольні питання:

► У чому полягає принципова відмінність між блоковими і поточковими шифрами?

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).

► Яким чином відбувається шифрування і дешифрування даних при використанні поточкових шифрів?

Випадковий потік бітів генерується по короткому секретному ключу за допомогою відкритого алгоритму, що називається генератором ключового потоку. Тут біти шифротекста розраховуються за правилом:

$$C_i = m_i \oplus k_i, \quad i = 0 \dots n$$

де $m_0, m_1 \dots$ – біти відкритого тексту;

$k_0, k_1 \dots$ – біти ключового потоку.

Дешифрування відповідно описується співвідношенням:

$$m_i = C_i \oplus k_i.$$

► Чи варто багаторазово використовувати ключі при поточковому шифруванні?

Припустимо, що повідомлення m_1 та m_2 були зашифровані одним ключем k . Тоді зломисник, перехопивши шифротексти, легко знайде суму по модулю 2 відкритих текстів :

$$C_1 \oplus C_2 = (m_1 \oplus k) \oplus (m_2 \oplus k) = m_1 \oplus m_2.$$

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

► Що таке генератор ключового потоку? Опишіть його основні властивості.

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.

► ***В чому полягає алгоритм установки ключа RC4?***

Секретний ключ задається набором чисел, які поміщаються в ключовий масив K , що також містить 255 елементів. Зазвичай вибирають коротку послідовність чисел, яка потім повторюється до заповнення K

► ***Опишіть алгоритм генерації псевдовипадкових чисел RC4.***

Генерують байти k псевдовипадкового ключового потоку, вибираючи випадкові елементи масиву S і змінюючи S для наступної вибірки:

- параметрам i, j присвоюється нульове значення;
- для генерації кожного байта випадкового потоку використовується

наступний алгоритм :

- $i = (i + 1) \bmod 256$;
- $j = (j + S[i]) \bmod 256$;
- $S[i]$ і $S[j]$ міняються місцями;
- $t = (S[i] + S[j]) \bmod 256$;
- $k = S[t]$.