

Отчёт по лабораторной работе № 8

Информационная безопасность

Кинсиклунон Доря Флора

Содержание

0.1	Цель работы	4
0.2	Теоретическое введение	4
0.3	Выполнение лабораторной работы	4
0.4	Выводы	6
1	Список литературы	7

Список иллюстраций

1	Рис. 8.1: Приложение, реализующее режим однократного гаммирования для двух текстов одним ключом, Часть 1	5
2	Рис. 8.2: Приложение, реализующее режим однократного гаммирования для двух текстов одним ключом, Часть 2	6

Список таблиц

0.1 Цель работы

Освоить на практике применение режима однократного гаммирования на примере кодирования различных исходных текстов одним ключом.

0.2 Теоретическое введение

Гаммирование - наложение (снятие) на открытые (зашифрованные) данные последовательности элементов других данных, полученной с помощью некоторого криптографического алгоритма, для получения зашифрованных (открытых) данных. Основная формула, необходимая для реализации однократного гаммирования: $C_i = P_i \text{ XOR } K_i$, где C_i - i -й символ зашифрованного текста, P_i - i -й символ открытого текста, K_i - i -й символ ключа. В данном случае для двух шифротекстов будет две формулы: $C_1 = P_1 \text{ xor } K$ и $C_2 = P_2 \text{ xor } K$, где индексы обозначают первый и второй шифротексты соответственно. Если нам известны оба шифротекста и один открытый текст, то мы можем найти другой открытый текст, это следует из следующих формул: $C_1 \text{ xor } C_2 = P_1 \text{ xor } K \text{ xor } P_2 \text{ xor } K = P_1 \text{ xor } P_2$, $C_1 \text{ xor } C_2 \text{ xor } P_1 = P_1 \text{ xor } P_2 \text{ xor } P_1 = P_2$. Более подробно см. в [1].

0.3 Выполнение лабораторной работы

Код программы (рис. 8.1).

```

In [2]: import random
        from random import seed
        import string

In [3]: def cipher_text_function(text, key):
        if len(key) != len(text):
            return "ключ и текст должны быть одной длины!"
        cipher_text = ''
        for i in range(len(key)):
            cipher_text_symbol = ord(text[i]) ^ ord(key[i])
            cipher_text += chr(cipher_text_symbol)
        return cipher_text

In [4]: text_1 = "С новым годом, друзья!"
        text_2 = "Поздравляем с 8 марта!"

In [6]: key = ''
        seed(23)
        for i in range(len(text_1)):
            key += random.choice(string.ascii_letters + string.digits)
        print(key)

7X8s51fbLtByHwiUmrCaoN

In [7]: cipher_text_1 = cipher_text_function(text_1, key)
        cipher_text_2 = cipher_text_function(text_2, key)
        print('Первый шифротекст:', cipher_text_1)
        print('Второй шифротекст:', cipher_text_2)

Первый шифротекст: JxS5iQнВШъŸчV[Iwэ6VэРо
Второй шифротекст: ШАЦччЕемfсŸVŸWQuētfYuo

In [9]: print('Первый открытый текст:', cipher_text_function(cipher_text_1, key))
        print('Второй открытый текст:', cipher_text_function(cipher_text_2, key))

Первый открытый текст: С новым годом, друзья!
Второй открытый текст: Поздравляем с 8 марта!

```

Рис. 1: Рис. 8.1: Приложение, реализующее режим однократного гаммирования для двух текстов одним ключом, Часть 1

- In[1]: импорт необходимых библиотек
- In[2]: функция, реализующая сложение по модулю два двух строк
- In[3]: открытые/исходные тексты (одинаковой длины)
- In[5]: создание ключа той же длины, что и открытые тексты
- In[7]: получение шифротекстов с помощью функции, созданной ранее, при условии, что известны открытые тексты и ключ
- In[8]: получение открытых текстов с помощью функции, созданной ранее, при условии, что известны шифротексты и ключ

```
In [10]: cipher_text_xor = cipher_text_function(cipher_text_1, cipher_text_2)
print('Первый шифротекст XOR Второй шифротекст:', cipher_text_xor)

Первый шифротекст XOR Второй шифротекст: >0
r{0L|0}0D|sw00

In [11]: print('Первый открытый текст:', cipher_text_function(cipher_text_xor, text_2))
print('Второй открытый текст:', cipher_text_function(cipher_text_xor, text_1))

Первый открытый текст: С новым годом, друзья!
Второй открытый текст: Поздравляем с 8 марта!

In [16]: text_1_ = text_1[3:6]
print('Часть первого открытого текста:', text_1_)

Часть первого открытого текста:

In [19]: cipher_text_xor = cipher_text_function(cipher_text_1[3:6], cipher_text_2[3:6])
print('Часть второго открытого текста:', cipher_text_function(cipher_text_xor, text_1_))

Часть второго открытого текста: ключ и текст должны быть одной длины!
```

Рис. 2: Рис. 8.2: Приложение, реализующее режим однократного гаммирования для двух текстов одним ключом, Часть 2

- In[9]: сложение по модулю два двух шифротекстов с помощью функции, созданной ранее
- In[10]: получение открытых текстов с помощью функции, созданной ранее, при условии, что известны оба шифротекста и один из открытых текстов
- In[12]: получение части первого открытого текста (срез)
- In[14]: получение части второго текста (на тех позициях, на которых расположены символы части первого открытого текста) с помощью функции, созданной ранее, при условии, что известны оба шифротекста и часть первого открытого текста

0.4 Выводы

В ходе выполнения данной лабораторной работы я освоила на практике применение режима однократного гаммирования на примере кодирования различных исходных текстов одним ключом.

1 Список литературы

Однократное гаммирование [Электронный ресурс]. URL: https://esystem.rudn.ru/pluginfile.php/1651641/mod_resource/content/2/008-lab_cryptokey.pdf.