

---

## Front matter

---

lang: ru-RU title: Отчёт по лабораторной работе №8 author: Георгес  
Гедеон institute: РУДН, Москва, Россия

date: 26 Октября 2024

## Formatting

---

toc: false slide\_level: 2 theme: metropolis header-includes:

- `\metroset{progressbar=frametitle,sectionpage=progressbar,numberi  
ng=fraction}`
- `'\makeatletter'`
- `'\beamer@ignorenonframefalse'`
- `'\makeatother' aspectratio: 43 section-titles: true`

---

# Отчет по лабораторной работе №7

---

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

---

### 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. Аналогичным образом можно найти ключ:  $K_i = C_i \text{ XOR } P_i$ . Необходимые и достаточные условия абсолютной стойкости шифра: • длина открытого текста равна длине ключа • ключ должен использоваться однократно • ключ должен быть полностью случаен

Более подробно см. в [1].

### Код программы(Рисунок 3.1).

```
In [6]: import random
        from random import seed
        import string
```

```
In [7]: 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 [8]: text_1 = "С новым годом, друзья!"
        text_2 = "Поздравляем с 8 марта!"
```

```
In [9]: key = ''
        seed(20)
        for i in range(len(text_1)):
            key += random.choice(string.ascii_letters + string.digits)
        print(key)
```

5URYX45jqR025g3uK5kbAA

```
In [10]: 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: ДѹАХѠЪЈТНОКЪКСНѢЮЎ  
Второй шифротекст: ЪжнИЕІёомѢУГѢОУѢСІРѨ

```
In [11]: 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!

```
{ width=70% }
```

In [ ]:

```
In [12]: cipher_text_xor = cipher_text_function(cipher_text_1, cipher_text_2)
print('Первый шифротекст XOR Второго шифротекста:', cipher_text_xor)
```

Первый шифротекст XOR Второй шифротекст: >0

$$r\{0 \mid 1\} \mid \text{sw}$$

```
In [13]: 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 [14]: text_1_ = text_1[3:6]
print('Часть первого открытого текста:', text_1_)
```

Часть первого открытого текста: овы

```
In [15]: 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: дра

```
{ width=70% }
```

• 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[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

## Выводы

---

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