Front matter

lang: ru-RU title: "Лабораторная работа №8" subtitle: "Дисциплина: Основы информационной безопасности" author: "Георгес Гедеон"

Formatting

toc-title: "Содержание" toc: true # Table of contents toc_depth: 2 lof: true # Список рисунков lot: true # Список таблиц fontsize: 12pt linestretch: 1.5 papersize: a4paper documentclass: scrreprt polyglossia-lang: russian polyglossia-otherlangs: english mainfont: PT Serif romanfont: PT Serif sansfont: PT Sans monofont: PT Mono mainfontoptions: Ligatures=TeX romanfontoptions: Ligatures=TeX sansfontoptions: Ligatures=TeX,Scale=MatchLowercase monofontoptions: Scale=MatchLowercase indent: true pdf-engine: lualatex header-includes:

- \linepenalty=10 # the penalty added to the badness of each line within a paragraph (no associated penalty node) Increasing the value makes tex try to have fewer lines in the paragraph.
- \interlinepenalty=0 # value of the penalty (node) added after each line of a paragraph.
- \hyphenpenalty=50 # the penalty for line breaking at an automatically inserted hyphen
- \exhyphenpenalty=50 # the penalty for line breaking at an explicit hyphen
- \binoppenalty=700 # the penalty for breaking a line at a binary operator
- \relpenalty=500 # the penalty for breaking a line at a relation
- \clubpenalty=150 # extra penalty for breaking after first line of a paragraph
- \widowpenalty=150 # extra penalty for breaking before last line of a paragraph
- \displaywidowpenalty=50 # extra penalty for breaking before last line before a display math
- \brokenpenalty=100 # extra penalty for page breaking after a hyphenated line
- \predisplaypenalty=10000 # penalty for breaking before a display
- \postdisplaypenalty=0 # penalty for breaking after a display

- \floatingpenalty = 20000 # penalty for splitting an insertion (can only be split footnote in standard LaTeX)
- \raggedbottom # or \flushbottom
- \usepackage{float} # keep figures where there are in the text
- \floatplacement{figure}{H} # keep figures where there are in the text

---

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: Ci = Pi XOR Ki, где Ci - i-й символ зашифрованного текста, Pi - i-й символ открытого текста, Ki - i-й символ ключа. Аналогичным образом можно найти ключ: Ki = Ci XOR Pi. 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].

Выполнение лабораторной работы

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)

        5URYX45jqRO25g3uK5kbAA
```

```
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: Дuѯѧѫѿљ JтѤѻќљК□сћѶќЮЎ`
         Второй шифротекст: ѢѫѥѩИЄЇёоѧѳ□VG□UѷЅЫРѱ`
```

```
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[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 [ ]:
```

```
In [12]: 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: >O

         r{ЭЛ|O}ЭД|swЭЭ
```

```
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[9]: сложение по модулю два двух шифротекстов с помощию 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.