# Лабораторная работа №7

Карымшаков А.А. - студент группы НФИбд-03-18

14.02.2022

# Элементы криптографии.

# Однократное гаммирование

# Прагматика выполнения

- Криптография - наука о методах шифрования. Знание однократного гаммирования и его особенностей является необходимым для дальнейшего знакомства с криптографией.

# Цель выполнения лабораторной работы

- Освоить на практике применение режима однократного гаммирования

# Задачи выолнения работы

- Написать программу, которая должна определить вид шифротекста при известном ключе и известном открытом тексте
- Также эта программа должна определить ключ, с помощью которого шифротекст может быть преобразован в некоторый фрагмент текста, представляющий собой один из возможных вариантов прочтения открытого текста

# Результаты выполнения лабораторной работы

- Написал программу, которая определяет вид шифротекста при известном ключе и известном открытом тексте (рис - @fig:001, рис - @fig:002)

Ввод [1]:
```
import numpy as np
```

Ввод [2]:
```
def encryption(text):
    print("Открытый текст: ", text)
    # Задам массив из символов открытого текста в шестнадцатеричном представлении:
    text_array = []
    for i in text:
        text_array.append(i.encode("cp1251").hex())
    print("\nОткрый текст в шестнадцатеричном представлении: ", *text_array)

    # Задам случайно сгенерированный ключ в шестнадцатеричном представлении:
    key_dec = np.random.randint(0, 255, len(text))
    key_hex = [hex(i)[2:] for i in key_dec]
    print("\nКлюч в шестнадцатеричном представлении: ", *key_hex)

    # Задам зашифрованный текст в шетснадцатеричном представлении:
    crypt_text = []
    for i in range(len(text_array)):
        crypt_text.append("{:02x}".format(int(text_array[i], 16) ^ int(key_hex[i], 16)))
    print("\nЗашифрованный текст в шестндцатеричном представлении: ", *crypt_text)

    # Задам зашифрованный текст в обычном представлении:
    final_text = bytearray.fromhex("".join(crypt_text)).decode("cp1251")
    print("\nЗашифрованный текст: ", final_text)
    return key_hex, final_text
```

**Рис. 1:** Функция, шифрующая данные

Ввод [4]:
```
# Изначальная фраза:
phrase = "С Новым Годом, друзья!"
# Получение сгенерированного ключа и зашифрованной фразы:
crypt_key, crypt_text = encryption(phrase)
```

```
Открытый текст:  С Новым Годом, друзья!

Открый текст в шестнадцатеричном представлении:  d1 20 cd ee e2 fb ec 20 c3 ee e4 ee ec 2c 20 e4 f0 f3 e7 fc ff 21

Ключ в шестнадцатеричном представлении:  bd 34 23 ce 18 10 89 50 12 92 62 c1 83 78 ac 54 2b 17 36 9b ef 1

Зашифрованный текст в шестндцатеричном представлении:  6c 14 ee 20 fa eb 65 70 d1 7c 86 2f 6f 54 8c b0 db e4 d1 67 10 20

Зашифрованный текст:  l□о ълерС|†/оТЊ°ЫдСg□
```

**Рис. 2:** Результат работы функции, шифрующей данные

- Написанная мною программа определяет ключ, с помощью которого шифротекст может быть преобразован в некоторый фрагмент текста, представляющий собой один из возможных вариантов прочтения открытого текста (рис - @fig:003, рис - @fig:004)

```
Ввод [3]: def decryption(text, final_text):
              print("Открытый текст: ", text)
              print("\nЗашифрованный текст: ", final_text)

              # Задам массив из символов открытого текста в шестнадцатеричном представлении:
              text_hex = []
              for i in text:
                  text_hex.append(i.encode("cp1251").hex())
              print("\nОткрый текст в шестнадцатеричном представлении: ", *text_hex)

              # Задам массив из символов зашифрованного текста в шестнадцатеричном представлении:
              final_text_hex = []
              for i in final_text:
                  final_text_hex.append(i.encode("cp1251").hex())
              print("\nЗашифрованный текст в шестнадцатеричном представлении: ", *final_text_hex)

              # Найду ключ:
              key = [hex(int(i, 16) ^ int(j, 16))[2:] for (i, j) in zip(text_hex, final_text_hex)]
              print("\nНужный ключ в шестнадцатеричном представлении: ", *key)
              return key
```

**Рис. 3:** Функция, дешифрующая данные

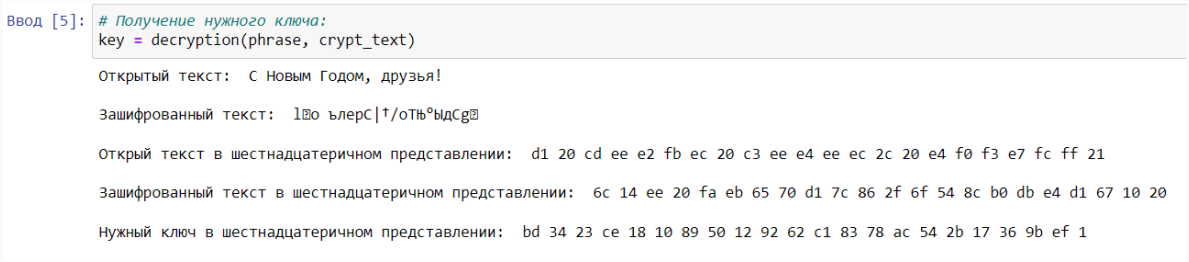


**Рис. 4:** Результат работы функции, дешифрующей данные

Ввод [6]:
```
# Проверка правильности ключа:
print("Ключ верен!") if crypt_key == key else print("Ключ неверен!")
```

```
Ключ верен!
```

**Рис. 5:** Сравнение ключей

Таким образом, я освоил на практике применение режима однократного гаммирования.